Aalto University
School of Chemical
Technology

School of Chemical Technology
Degree Programme of Chemical Technology

Ella Potka

MANAGEMENT OF COMPLEX DATA SETS IN PROCESS INDUSTRY

Master's thesis for the degree of Master of Science in Technology submitted for inspection, Espoo, 13.2.2017.

Supervisor **Professor Sirkka-Liisa Jämsä-Jounela**

Instructor **M.Sc. Lauri Haapanen**
M.Sc. Markus Sintonen

Aalto University
School of Chemical
Technology

Aalto University, P.O. BOX 11000, 00076 AALTO

www.aalto.fi

Abstract of master's thesis

Author Ella Potka

Title of thesis Management of complex data sets in process industry

Department Department of Biotechnology and Chemical Technology

Professorship Process Control **Code of professorship** KE-90

Thesis supervisor Professor Sirkka-Liisa Jämsä-Jounela

Thesis advisor(s) / Thesis examiner(s) M.Sc, Lauri Haapanen, M.Sc. Markus Sintonen

Date 13.2.2017 **Number of pages** 86 **Language** English

Abstract

The idea of Internet of Things (IoT) is to connect all the devices into one network and to enable interoperability between them. Interoperability benefits also the process industry when the control devices and software can interoperate with management software. One part of the industrial IoT is being able to efficiently analyze the data from the field devices so that for example predictive maintenance can be achieved. Information modelling is needed to enable communication between the different software and to make analyzing data easier. This thesis examines the state of the IoT and the benefits of information modelling. The aim is to find the information modelling standard most suitable for the process industry and to figure out how standard conforming information models are created.

The literature part of this thesis studies the current state and the future of IoT. The focus is especially on the possibilities it brings for the oil and gas industry. A broad collection of information modelling standards is introduced. According to the comparison made, OPC UA was selected in this work as the most suitable standard for the needs of process industry.

In the experimental part the information modelling process is introduced and three OPC UA modelling tools are examined. Instructions for information modelling with OPC UA were created. An OPC UA standard conforming information model of a distillation column was created to be used to configure a soft sensor. The model was validated using expert knowledge. The model was also successfully connected to a data source that was in this case a DCS emulator.

Keywords Information modelling, OPC UA, Industrial Internet of Things

Aalto-yliopisto
Kemian tekniikan korkeakoulu

Aalto-yliopisto, PL 11000, 00076 AALTO

www.aalto.fi

Diplomityön tiivistelmä

Tekijä Ella Potka

Työn nimi Suurten datamäärien hallinta prosessiteollisuudessa

Laitos Biotekniikan ja kemian tekniikan laitos

Professuuri Prosessien ohjaus **Professuurikoodi** KE-90

Työn valvoja Professori Sirkka-Liisa Jämsä-Jounela

Työn ohjaaja(t)/Työn tarkastaja(t) DI Lauri Haapanen, DI Markus Sintonen

Päivämäärä 13.2.2017 **Sivumäärä** 86 **Kieli** Englanti

Tiivistelmä

Esineiden internetin ajatuksena on kytkeä kaikki laitteet samaan verkkoon ja mahdollistaa niiden välinen yhteensopivuus. Myös prosessiteollisuudessa on hyötyä yhteensopivuudesta, kun säätölaitteet ja ohjausjärjestelmät voivat kommunikoida hallintojärjestelmien kanssa. Teollisessa esineiden internetissä kenttälaitteiden tuottamaa data pystytään analysoimaan tehokkaasti siten, että esimerkiksi ennakoiva huolto on mahdollista. Tietomalleja tarvitaan laitteiden välisen kommunikaation mahdollistamiseksi ja tiedon analysoinnin helpottamiseksi. Tämä diplomityö käsittelee esineiden internetin tilaa sekä tietomallinnuksella saavutettavia hyötyjä. Tavoitteena on löytää prosessiteollisuuteen sopivin tietomallinnusstandardi sekä selvittää, miten valitun standardin mukaisia tietomalleja laaditaan.

Kirjallisuusosassa selvitellään esineiden internetin nykytila sekä tulevaisuudennäkymät. Erityisest keskitytään esineiden internetin öljy- ja kaasuteollisuudelle tuomiin mahdollisuuksiin. Työssä esitellään laaja kokoelma tietomallinnusstandardeja. Tehdyn vertailun jälkeen OPC UA valittiin tässä työssä prosessiteollisuuden käyttötarkoitukisiin sopivimmaksi standardiksi.

Soveltavassa osassa esitellään tietomallinnusprosessi sekä tutustutaan kolmeen erilaiseen OPC UA tietomallinnustyökaluun. Tietomallintamisesta OPC UA -standardin avulla laadittiin ohjeet. Työssä laadittiin OPC UA:n mukainen tietomalli tislauskolonnista virtuaalisen säätimen konfigurointikäyttöön. Laaditun mallin toimivuutta arvioitiin asiantuntijoiden avulla. Malli kiinnitettiin onnistuneesti tietolähteeseen, joka tässä tapauksessa oli DCS emulaattori.

Avainsanat Tietomallinnus, OPC UA, teollinen esineiden internet

PREFACE

This master's thesis was written in the Technology and Product Development department of Neste Jacobs Oy between 1st of April and 30th of September 2016.

First of all, I would like to thank Professor Sirkka-Liisa Jämsä-Jounela for supervising this thesis. I also wish to thank my advisors Lauri Haapanen and Markus Sintonen for the invaluable guidance and advices they gave me. Their input and precise comments were greatly appreciated. I want to show gratitude to Samuli Bergman and Antti Räisänen for sharing their expert knowledge and for their help during the practical part of the thesis. In addition to the above mentioned my other colleagues deserve a thank you for being cheerful and always giving me the best start to the day.

I want to express deep gratitude to my family for supporting me during my studies. Huge thanks to my friends for all the stupid things we have done and the great memories we have made. My free-time wouldn't have been as enjoyable without Laos-pellet, #Vauva.fi (not the website), 176, JTM3 and finally TPJTMK. I would like to express special gratitude to Kaisa for being almost like my personal study advisor during my master's studies. The project is finally done. Feels good.

Espoo, 2.1.2017

Ella Potka

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

ADI	Analyzer Device Integration
AML	AutomationML, Automation Markup Language
ARM	Architectural Reference Model
B2MML	Business to Manufacturing Markup Language
CAEX	Computer Aided Engineering Exchange
CIM	Common Information Model
COM	Component Object Model
DCOM	Distributed Component Object Model
DDS	Data Distribution Service
DI	Device Integration
DLL	Dynamic-Link Library
ERP	Enterprise Resource Planning
FDI	Field Device Integration
IIC	Industrial Internet Consortium
IIoT	Industrial Internet of Things
IoT	Internet of Things
IoT-A	Internet of Things - Architecture
IEC	The International Electrotechnical Commission
IIRA	Industrial Internet Reference Architecture
IOMOG	Assetricity Integrated Operations and Maintenance for Oil & Gas

ISA-95	International Society of Automation standard 95
ISBM	Information Service Bus Model
ISO	International Organization for Standardization
MES	Manufacturing Execution System
MIMOSA	Machinery Information Management Open Systems Alliance
OPC	Object Linking and Embedding for Process Control
OPC UA	OPC Unified Architecture
OWL	Ontology Web Language
O&G	Oil and Gas
P&ID	Piping and Instrumentation Diagram
RAMI 4.0	Reference Architectural Model for Industrie 4.0
RFID	Radio-Frequency Identification
SCADA	Supervisory Control and Data Acquisition
SCL	Substation Configuration Language
SFC	Sequential Functional Chart
SOA	Service-Oriented Architecture
UML	Unified Modelling Language
XML	Extensible Markup Language

1 INTRODUCTION

1.1 Background

Internet of Things (IoT) means connecting different physical objects with smart sensors to communicate with each other over the Internet. (Breivold, Sandström 2015) The industrial counterpart of IoT, the Industrial Internet of Things (IIoT), can provide the industry with effective predictive maintenance, equipment monitoring and resource optimization just to name a few possibilities. (Slaughter, Bean & Mittal 2015) IoT still needs a lot of technology development and changes in the infrastructure. To enable single system to handle dynamic business and engineering processes it is necessary to connect the existing factory automation systems with enterprise resource planning (ERP) and manufacturing execution systems (MES) over the IoT infrastructure. The challenge in this is the variety of proprietary control systems in the industry. (Kumar, Bose 2015) There are various efforts to standardize the interaction. Deciding between the different standards is hard because there is no tracking of the use of standards. Many initiatives like Germanys Industrie 4.0 and USA's Industrial Internet Consortium are contributing to standards. (Lu, Morris & Frechette 2016)

This master's thesis focusses on standards regarding information models and management of process data. Information modelling is a concept for presenting process data in a technologically independent way and providing interoperability. This is needed since automation systems use different technologies and standards that have their own ways to represent process data. A uniform view of the system is required for enterprise and management level applications, like ERP and MES. Having all the information of different automation systems in one unified management system makes cross-domain optimization possible. (van der Linden, Granzer & Kastner 2011)

There is a number of interesting standards for information modelling and exchange. This thesis studies a set of standards. It is not exhaustive but tries to cover a wide range of standards suitable for the industry's needs. Standards selected are OPC UA, ISA-95, ISO 15926, CAEX, AutomationML, PandIX, Common Information Model and IEC 61850. Because the variety of standards,

some initiatives on synchronization between the standards exist. One example of them is the Oil & Gas Interoperability Pilot. (Johnston et al. 2012) This thesis tries also to describe the synchronization and interoperability possibilities of standards when addressing them.

1.2 The objective

The first objective of this thesis is to take a look at the current state of Internet of Things and study the trends concerning oil and gas industry. Also the possibilities the IoT brings in the future are to be studied.

Another aim is to study how standards are used for information modelling and what are the benefits and drawbacks of different standards. The second question is what can be achieved with these information models and how the selection of the standards affects to this. Also the different stakeholders need to be studied to map their needs and expectations for the information models. The ultimate goal is to figure out the most important development needs and select the most suitable way to create the information models with these requirements in mind. The information modelling tools are studied and evaluated to be able to create the actual information models.

The main objective is to create information model of a distillation process unit. The created model needs to match the development needs defined for information models in usual applications and the ones defined specially for the use case. Also a tool to carry out this task is selected based on the evaluation made and based on what is supported in the server used for hosting the models in this thesis. The equipment modelled and the use case of the information model needs to be studied carefully before the modelling.

The results of the thesis present the benefits and the drawbacks of the standards studied as well as the guidelines to the process of information modelling. The reasoning is given for selection of the most suitable standard for the needs of process industry. An information model of a distillation process unit is done using the modelling guidelines written. The model is evaluated against the goals set for it. Also the results of comparison of modelling tools created for this standard are explained.

1.3 Structure of the thesis

The thesis starts with background check to the state of Internet of Things (IoT). The standards, trends and the possibilities are reviewed. The different standards and their applications are presented. In Section 3, information modelling and the existing modelling tools are introduced. The expectation and the needs of stakeholders and what is achieved with the information models is determined based on the knowledge gathered. The modelling practices and a comparison of modelling tools are presented in Section 4. The equipment to be modelled and the tools used are introduced in Section 5. Section 6 consists of explaining the modelling process while the models are explained in Section 7. Finally the conclusions and future work are discussed in Section 8.

2 INTERNET OF THINGS

This chapter provides a background on Internet of Things (IoT). First the future visions as well as the challenges concerning IoT are discussed. A brief overview on the standard organizations and initiatives is given in the second subchapter. Also the architectures driven by the different initiatives are introduced. Finally the trends specific to oil and gas industry are studied. Challenges and possible solutions are presented together with an example of the work aiming towards achieving interoperability in the industry.

2.1 Current state of Internet of Things

There is no single universal definition for the Internet of Things (IoT). In brief it means that objects with sensing, processing and identifying capabilities are all connected to the same network and communicate with each other. (Whitmore, Agarwal & Xu 2014) In 2015 IEEE defined IoT as “a self-configuring and adaptive system consisting of networks of sensors and smart objects whose purpose is to interconnect all things, including every day and industrial objects, in such a way as to make them intelligent, programmable and more capable of interacting with humans”. (Breivold, Sandström 2015) Currently IEEE is asking for comments to form a broader and more accurate definition of IoT. (IEEE 2015)

Whitmore et al. (Whitmore, Agarwal & Xu 2014) divided the most common applications of IoT into four sub-categories that are smart infrastructure, healthcare, supply chain/logistics and social applications. Smart infrastructure means connecting smart objects into physical infrastructure. An example of this is a smart grid that collects data about energy consumption and makes it available online for further analyzing. Smart infrastructures improve flexibility, reliability and efficiency of the infrastructure. In healthcare sector IoT could be used to automate some tasks that the patients have to perform. One scenario is placing sensors on health monitoring devices to collect information about the patient’s current health status. The data from the sensors could then be made available for doctors over the Internet to enable more efficient treatment. In the field of logistics, supply chains already use sensor networks in assembly lines and Radio-Frequency Identification (RFID) to track products. IoT can still provide

more detailed and up-to-date information further improving efficiency. Social applications of IoT include connecting IoT devices to services such as Facebook or Twitter. The devices could for example provide information about user's location and activities attended. This information could be used to tell the user if they are nearby a friend or some event that might interest them. (Whitmore, Agarwal & Xu 2014)

The biggest challenges of IoT are information security, data integrity and privacy, interoperability and scalability. Industrial IoT (IIoT) shares the challenges of consumer IoT, but of course there are also additional challenges like exact timing and criticality of the systems. Automation systems have to be accurate in time, reliable and safe. (Breivold, Sandström 2015)

Security issues grow larger as the systems grow larger and more complex. Encryption is seen as the key for secure information exchange. The current encryption algorithms are made for devices where the resources are not restricted. The smallest IoT devices however are limited in power and are currently unable to support robust encryption. The algorithms have to be made faster and less energy-consuming to enable encryption on the IoT. Moreover, efficient key distribution scheme should be found. (Bandyopadhyay, Sen 2011) Also identity is important factor for security. When communicating with smart devices, we have to be able to ensure that the device is what it claims to be. (Whitmore, Agarwal & Xu 2014) Additionally in industrial systems, the security updates have to be made without interference to the control of the process (Breivold, Sandström 2015).

IoT is basically a complex network of devices and software. Also in automation systems there are thousands of different components like controllers, workstations and servers. It means having a huge heterogeneity in interfaces and communication solutions. They all present and interpret data differently. There are efforts to standardize the communications to achieve interoperability. Integration of different systems is costly without standards. (Breivold, Sandström 2015)

Middleware is getting more and more important since it can well be used for integration of legacy technologies into new systems. In the last years the architecture proposed for middleware has been service-oriented architecture (SOA). The commonly mentioned advantages of SOA are enhanced flexibility, interoperability and reuse.(Atzori, Iera & Morabito 2010) SOA is a software architecture design in which the independent processes are usable via services. Services are some small functions such as reading or writing. The loose coupling of services enables the flexible interconnection between applications. (Rouse 2014)

2.2 IoT standard organizations and initiatives

The challenges in integrating systems have resulted in efforts to standardize the communications in automation systems. (Breivold, Sandström 2015) International standard organizations like ISO and IEC are working on standards for industrial process control and automation. ISO's committee on automation systems and integration (TC184) has two subcommittees, SC4 and SC5, that are particularly concerned on data exchange standards. SC4 focusses on industrial data standards such as ISO 10303 for exchange of product manufacturing information and ISO 15926 for integration of life-cycle data for process plants. SC5 on the other hand focusses on interoperability, integration and architectures for automation applications. IEC has developed standards like IEC 62264 which is standardized version of ISA-95 for integrated enterprise and control systems. There are also several consortia, like the OPC Foundation, developing standards for IIoT and communication between device and software. These sometimes are offered to ISO or IEC to achieve wider usage. Furthermore, professional societies like ISA and academically oriented societies like IEEE are working on standards. (Lu, Morris & Frechette 2016) The selection of organizations above is wide but not exhaustive. There are still many more organizations developing standards.

Standards are the enablers of efficient manufacturing systems by providing a method to exchange data between software and devices of different vendors. Interoperability between standards is still a problem. Because of the multiplicity of standards, there is a huge amount of obsolete standards. There is no tracking on the adoption of the standards. The industry and the software and device providers are on their own when trying to find the most suitable and most

common standards. The vast amount of the standard organizations also causes overlap and redundancy between the standards. The standards in the same technical area are be defined separately for different industry sectors. These standards are not always consistent which also causes overlapping and redundancy. To overcome these issues, the standard organizations have to collaborate for harmonizing standards. (Lu, Morris & Frechette 2016) According to ISO this means that different standards on the same subject provide similar information or interchangeability of products or processes. The presentation and the guidance on how to match the requirements can be different, but the output has to be the same. (ISO/IEC Guide 2:2004: 2004)

IoT has also given rise to new initiatives that contribute to the standards and create reference architectures. Next, four of the initiatives, Plattform Industrie 4.0, Industrial Internet Constortium, Arrowhead and Internet of Things - Architecture are presented. In addition to these, IEEE is developing Standard for an Architectural Framework for the Internet of Things. (Weyrich, Ebert 2016)

2.2.1 Plattform Industrie 4.0

Industrie 4.0 was originally a strategy developed by the German government to promote the computerization of manufacturing. Now Industrie 4.0 can be seen as an alternative expression of Internet of Things. In this thesis IoT is used meaning the intelligent and connected network of things and Industrie 4.0 is used only in this subchapter meaning the German alternative of IoT.

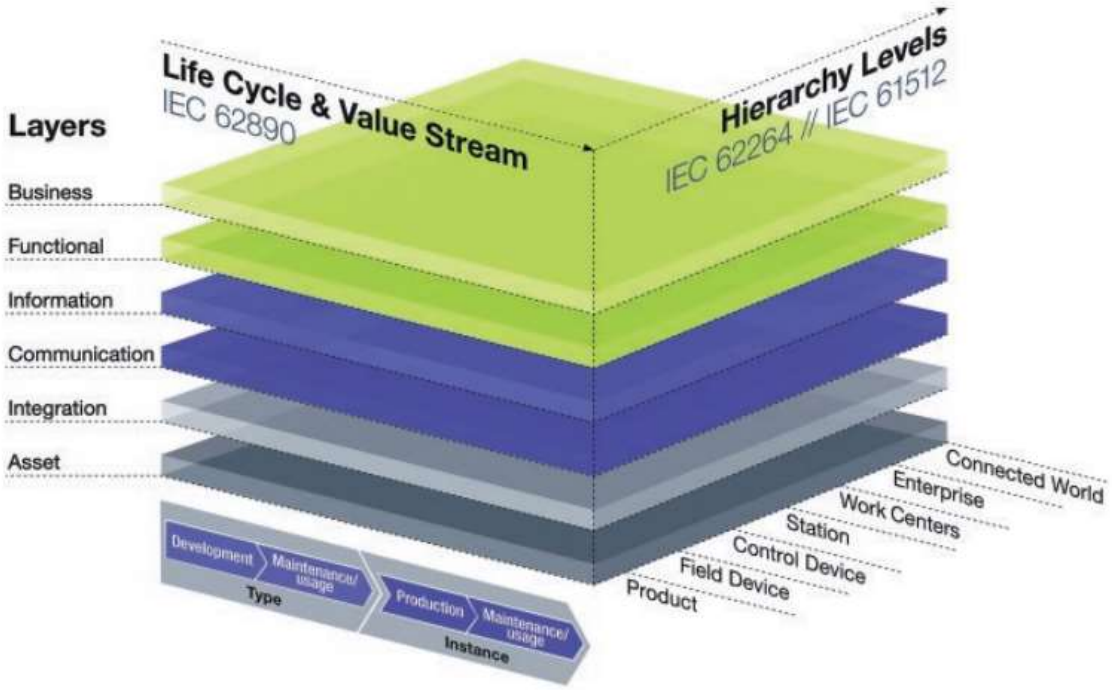

Figure 1. Reference architecture model for Industrie 4.0 (Adolphs et al. 2015)

Plattform Industrie 4.0 aims to develop a basis for a consistent and reliable IoT framework through a dialog with business, science and government. (Plattform Industrie 4.0 2016) The potential of Industrie 4.0 is in smart factories that allow meeting individual customer requirements, dynamic business and engineering processes, optimized decision making and new ways of creating value. In order to deliver these goals, Industrie 4.0 should be able to implement full integration of all the manufacturing systems. The value networks should be horizontally integrated as well as the whole automation pyramid vertically integrated meaning networked manufacturing systems. The entire value chain should be integrated from end to another. The Industry 4.0 Working Group has divided the work into eight key areas of which one, standardization and reference architecture, is in particular interest in this thesis. (Kagermann, Wahlster & Helbig 2013)

The result of the work is Reference Architectural Model for Industrie 4.0 (RAMI 4.0). (Plattform Industrie 4.0 2016) The first version was released in July 2015 by

ZVEI, VDMA and BITKOM. The reference architecture model is three dimensional. The “Hierarchy Levels” axis represents the different functionalities within factories. It uses hierarchies from IEC 62264 standard. The “Life Cycle & Value Stream” axis represents the life cycle of facilities and products and is based on IEC 62890. The “layers” axis describes the decomposition of a machine into its properties. The layers are business, functional, information, communication, integration and asset. The data models used for layers need to be consistent during the whole lifecycle and all hierarchy levels. The three dimensional model of Industrie 4.0 is presented in Figure 1. RAMI 4.0 is based on many existing standards. It proposes certain standards to be used in certain layers. In communication layer, OPC UA is used. For the Information layer IEC Common Data Dictionary, eCl@ass characteristics, Electronic Device Description (EDD) and Field Device Tool (FDT) are used. Field Device Integration (FDI) is used for implementation of functional and information layer. Finally for end-to-end engineering AutomationML, ProSTEP iViP and eCl@ass are used. (Adolphs et al. 2015)

2.2.2 Industrial Internet Consortium

The U.S. equivalent of Industrie 4.0 is the Industrial Internet Consortium (IIC) founded by GE, IBM, Cisco, Intel and AT&T. It brings together organizations and technologies necessary to accelerate the growth of Industrial Internet. The consortium has different committees, such as the technology committee. In case of this thesis, the architecture task group of the technology committee is in particular interest. It has developed a reference architecture called the Industrial Internet Reference Architecture (IIRA). (Industrial Internet Consortium 2016)

The IIRA documentation defines an Industrial Internet System (IIS). The IIS is a large network connecting industrial control systems to people and integrating them with other systems such as enterprise systems. According to IIRA the IIS has various concerns that can be grouped as viewpoints. These viewpoints are business, usage, functional and implementation. The business viewpoint is concerned with the identification of business stakeholders. The usage viewpoint focuses on the expected system usage. The functional viewpoint addresses functional components of IIS, their interrelation and external interactions. Finally,

the implementation viewpoint deals with the technologies needed. (Industrial Internet Consortium 2016)

When discussing the architecture itself the functional and implementation viewpoints are the points of interest. The IIS is decomposed into five functional domains which are control, operations, information, application and business domains. Information is exchanged between these domains. Figure 2 shows the relations of functional domains and the data and control flows between them. The architecture is described in the implementation viewpoint. The architecture patterns suggested include the three-tier pattern and the gateway-mediated edge connectivity and management pattern. (Industrial Internet Consortium 2016)

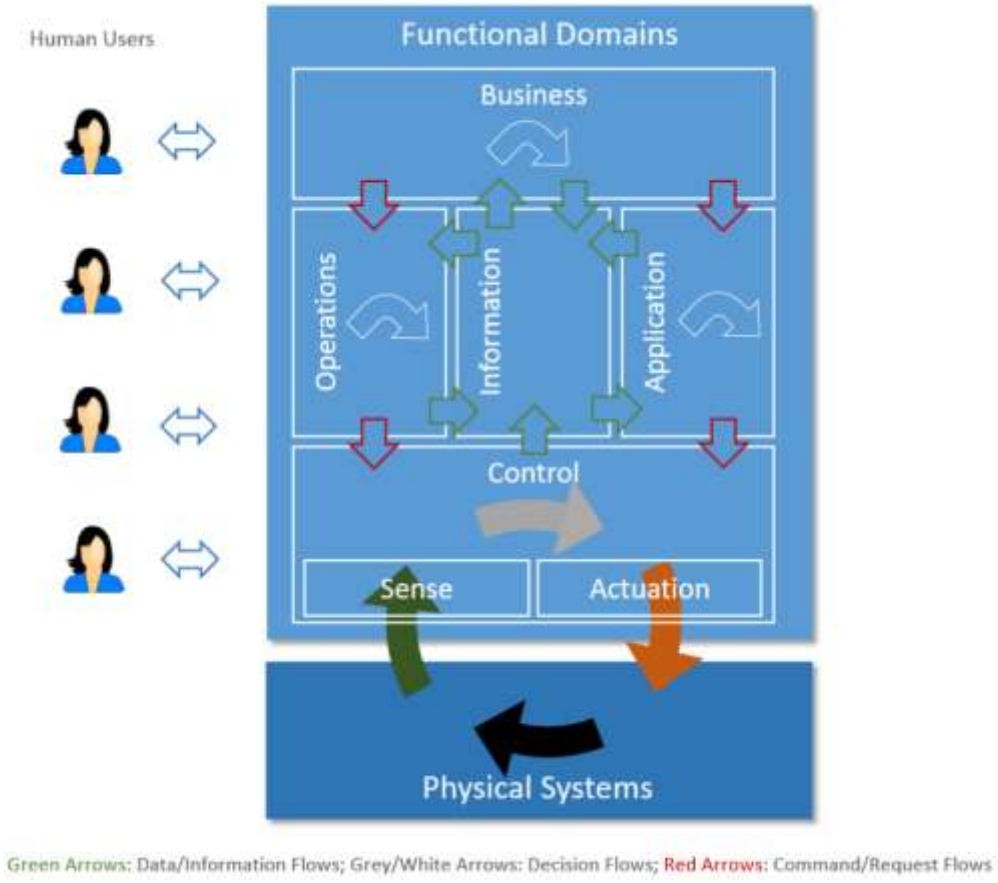

Figure 2. The functional domains of IIRA. (Industrial Internet Consortium 2016)

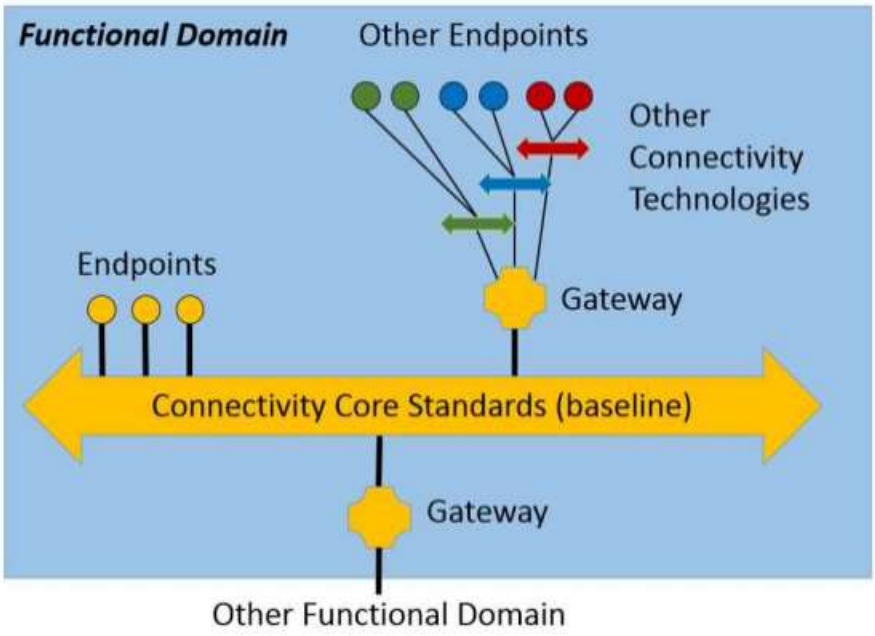

Figure 3. Industrial Internet Reference Architecture (Industrial Internet Consortium 2015)

To integrate all the different technologies IIRA suggests a concept where all the subsystems are connected with a core connectivity "databus". This can be seen in Figure 3. Some subsystems may require a gateway for connection and transition from data standard to another. The "databus" or a Connectivity Core Standards as IIRA calls it needs standards that fulfill certain requirements. The requirements are achieving interoperability between endpoints, automated service discovery, performance and scalability, programming model, Quality of Service (QoS) and support to peer-to-peer, client-server and publish-subscribe patterns. (Prismtech (c) 2016) The QoS parameters are reliable data delivery, timeliness, ordering, durability, lifespan, fault tolerance and security. In the first release IIRA doesn't recommend a specific standard to be used in the "databus". (Industrial Internet Consortium 2015) However, Real-Time Innovations (RTI) sees Data Distribution Service (DDS) by the Object Management Group (OMG) as the best core connectivity standard for the architecture. (Schneider 2015)

2.2.3 Arrowhead

The Arrowhead is a project funded by the European Union. The vision of the project is to enable interoperability of services provided by almost any device. The project aims to provide technical framework and propose solutions for integration with legacy systems. The technical solution is evaluated with real experimentations in different industry sectors. The project targets five business domains which are production, smart buildings and infrastructures, electro mobility, energy production and virtual markets of energy. (Arrowhead 2016a)

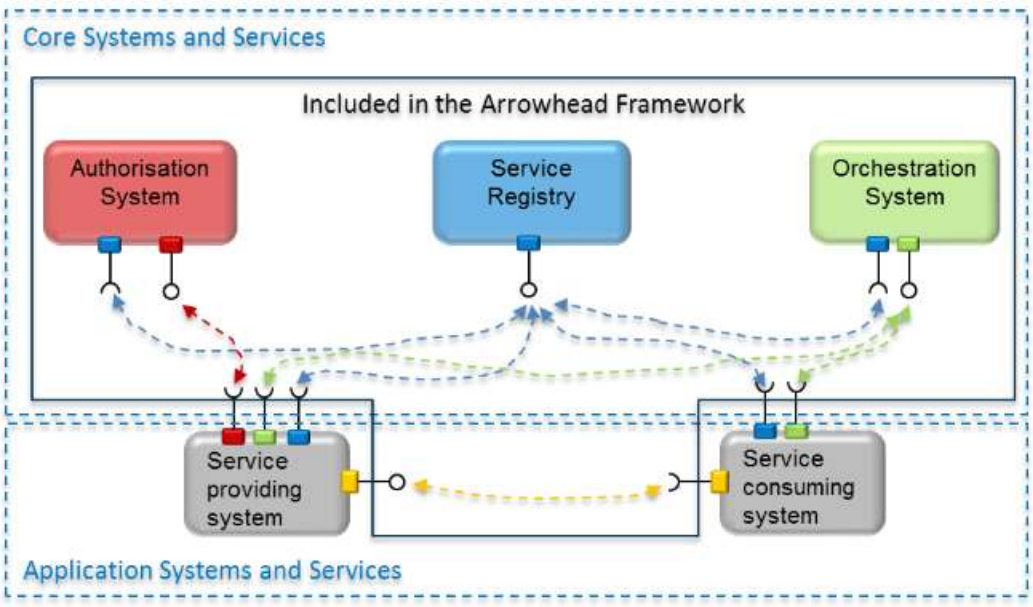

Figure 4. The Arrowhead Framework (Arrowhead 2016b)

The Arrowhead Framework is a SOA based framework for integrating multi-vendor applications. The framework consists of Core Services and Application Services. These can be seen in Figure 4. The Application Services handle the exchange of information while the Core Services support them. The specialized Application Services could be for example reading sensors. The Core Services for example provide application installation or status monitoring functionality. The framework addresses also design of gateways or mediators that make systems with different standards compliant with Arrowhead. (Arrowhead 2016b)

2.2.4 Internet of Things - Architecture

Internet of Things – Architecture (IoT-A) was another project funded by the European Union to develop a common reference model and architecture for IoT. It was active from 01.09.2010 to 31.08.2013. Like the other initiatives, also IoT-A aimed to the interoperability of different solutions. It based the development on the existing standards. The architecture developed in the project is called Architectural Reference Model (ARM). (Bauer et al. 2013b)

The ARM consists of three parts. The first one is the IoT Reference Model that provides the highest abstraction level for ARM. The IoT Reference Model provides an IoT Domain Model which is the top-level description of the architecture. Other relevant models are the sub-models that address the information, functional, communication and the security views. The Reference Architecture is the second part of ARM and the reference for building compliant IoT architectures. The architecture consists of views that build further on the models defined in IoT Reference Model. The last part of ARM is the guidelines on how to derive a concrete architecture from the model. (Bauer et al. 2013a)

2.2.5 Comparison of Internet of Things architectures

The four architectures for IoT have different perspectives. IoT-A provides a detailed view of the information technology related to IoT. IIRA is strongly focused on industry but also includes healthcare, energy and transportation information. RAMI4.0 on the other hand focusses on manufacturing and logistics details. (Weyrich, Ebert 2016) Arrowhead's goal is to build architecture for automation in production, buildings, electro-mobility and energy-market. (Arrowhead 2016a) The different perspectives lead to architectural differences, for example the presentation of semantics. Because IIRA focuses on industry, also the data description is focusing on functionality of that domain. RAMI4.0 is almost the same as IIRA, but includes additional life-cycle and value stream data. (Weyrich, Ebert 2016) Arrowhead is concentrated on industry, business and energy data. (Arrowhead 2016b) IoT-A is more generic when it comes to the semantics. IoT-A has also broader definitions for middleware functionality and cloud aspects while IIRA addresses the same things but is more focused on business and use cases. (Weyrich, Ebert 2016)

In March 2016 IIC and Plattform Industrie 4.0 agreed to work together to see if it is possible to achieve interoperability and alignment of IIRA and RAMI4.0. (Federal Ministry for Economic Affairs and Energy 2016b, RTI News 2016) There is however no desire to merge the two architectures since they focus on different domains. As said earlier, RAMI4.0 has focusses on manufacturing in depth while IIRA is more cross domain focused. The industry however needs to be able to operate cross domain, manufacturing being one of the domains. The domains are illustrated clearly in Figure 5. (Federal Ministry for Economic Affairs and Energy 2016a)

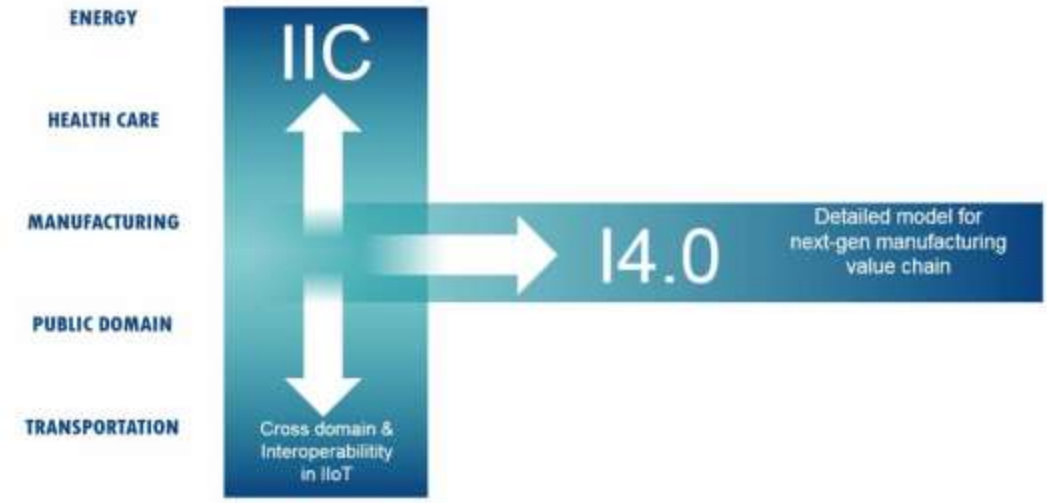

Figure 5. The domains the IIRA (IIC) and RAMI4.0 (I4.0) operate on. (Federal Ministry for Economic Affairs and Energy 2016a)

On 13th of April OMG and OPC Foundation announced to collaborate for interoperability of the underlying DDS (IIRA) and OPC UA (RAMI4.0) standards. The organizations have found two ways to achieve interoperability and are developing them. The first is a “OPC UA/DDS gateway” that allows applications and devices using DDS to connect to OPC UA and vice versa. The second is “OPC UA DDS Profile” which enables integrated use cases. (Object Management Group 2016)

2.3 IoT trends concerning oil and gas industry

Because of the low market prices of oil and the cost of producing oil going up, the oil and gas (O&G) industry today is in need of enhanced production technologies. Each step of the chain from drilling to customer has to be made more efficient. IoT is seen as one solution, because it provides a way to gather and connect information about processes, supply chains and customer relationships. In addition to optimizing the mentioned things, the new information can bring innovative aspects on how the ways to do business should be changed. Because the O&G industry is so diverse there is no single IoT solution for all. The objectives for everyone, however, are more or less the same. The common goals are to improve reliability of processes, optimize operations and create new value. (Slaughter, Bean & Mittal 2015)

Traditionally the oil companies have been looking for technologies to improve single discipline, for example to exploit more complex resources. The investments go to a new process or control system when needed or to development of a single technology. A so called silo effect is caused by giving no attention to integrating these new systems with existing ones. This means that different organizations of the company or parts of the factory are separated disciplines with distinct roles. The silos limit the agility and cripple IoT. The organizations, or silos, have to be connected and interoperating to get real benefits from IoT. (Moriarty et al. 2015)

Instead of focusing on individual technologies, more value can be obtained when the new technologies are integrated cross disciplines. Only a minor part of the data gathered from the refinery processes is available for the industry's decision makers. Increasing this availability and analysis can save money by for example eliminating unplanned outages. (Slaughter, Bean & Mittal 2015)

According to survey made by Cisco (Moriarty et al. 2015) data is the area of IoT where the O&G leaders see the need for improvements. Cisco sees three key challenges in data area of IoT and proposes solutions for them. As a solution for integrating heterogeneous data from distributed sources the data is virtualized. The data can actually be stored anywhere, but it seems to origin from one source. The second challenge is automating the data collection to get the data to the right place at the right time to be analyzed. Sometimes, for example in the

offshore drilling-platforms, the data cannot be moved, because of weak connections. With smart devices the data can be processed at the edge of the network. The third problem is the lack of skills or resources to analyze the data gathered. Solving this requires attracting employees with sufficient knowledge.

New connected sensors and closed loop systems, where changes to operation are made by machines pose new vulnerabilities. This creates more opportunities for cybercriminals to exploit. Many of the oil and gas companies do not have a proper response plan for cybersecurity incidents yet there has been many attacks targeting the energy sector. This has to change before the companies can fully take advantage of IoT. (Moriarty et al. 2015)

According to Cisco Consulting Services (Moriarty et al. 2015), IoT has potential to create over 500 million dollars of net profit for an oil company with 50 billion dollar revenue and production of 270 million barrels annually. Most, 83%, of this profit comes from improvements in upstream operations, while midstream and downstream upgrades are only minor part. The analysis, however, doesn't take into account how the implementation costs divide between the different operations. The analysis clearly pointed out the importance of data. The biggest profit producing operation was reducing lifting and production costs, where the value comes from better monitoring and data management capabilities, real time optimization and automatic analyses. The second biggest value generator was rig uptime, which depends on advanced sensors and Big Data analytics to conduct predictive maintenance. (Moriarty et al. 2015)

In the upstream sector, or the exploration and production sector, the technological complexity is increasing. This means installing new sensors which produce a bigger flow of data. In addition the scale and the frequency of the data are growing and there is a need to expand the scope. These data flows cannot be fully taken advantage of because of the weak data-management capabilities. The communications between different software is limited by the lack of open standards and the diversity and incompatibility of the proprietary communication formats. Overcoming these challenges creates possibilities like automating the production, faster deployment of new projects and better modelling of the earth's surface to find oil. (Slaughter, Bean & Mittal 2015)

In midstream, the shale oil boom, has made transporting the liquids and natural gas more complex, since the volumes and locations have started altering rapidly. The old pipelines and monitoring devices are causing losses due to fuel leaks. The safety and reliability could be improved with investing to new sensor technologies. Further analyzing the data from these sensors could benefit in better selection of shipping routes. (Slaughter, Bean & Mittal 2015)

The most mature part and therefore also financially the most challenging part of O&G industry is refining crude-oil. A critical part with potential for improvement is avoiding unscheduled shutdowns. At the moment the maintenance done for the equipment is time-based preventive planning where the equipment is taken to workshop for inspection without knowledge of the actual condition. Time is wasted inspecting equipment that isn't in the need of repairing. With new sensors technologies, advanced wireless networks, open standards and integrated device management the strategy can be shifted to condition-based predictive maintenance. Another problem in downstream O&G is that so far the information has been analyzed mostly on the plant level only. If this scope is made wider the whole supply chain and the logistics after production can be taken into account. One benefit from this kind of data analyzing could be the ability to buy crude oil dynamically from various sources instead of long contracts. (Slaughter, Bean & Mittal 2015)

As described earlier, the amount of data gathered in the upstream sector is growing but the traditional SCADA systems use proprietary protocols which hinder the exchange of data. Also the governments have started demanding reports about the drilling conditions and safety in a standardized form. This has led the petroleum industry to drive towards standardization of data exchange. (Cotton et al. 2012) Standards Leadership Council (SLC) was formed in 2012 to unite the standard organizations of upstream O&G to promote the adoption of open standards. The consortium has several member organizations including The OPC Foundation, Energetics and POSC Caesar Association. (Standards Leadership Council 2016) Norway has been one of the leaders in the upstream O&G standardization. In 2008-2012 a project called The Integrated Operations in the High North Joint Industry Project (IOHN) tested using ISO 15926 to ensure interoperability, to facilitate integration and to transfer data. (Cotton et al. 2012)

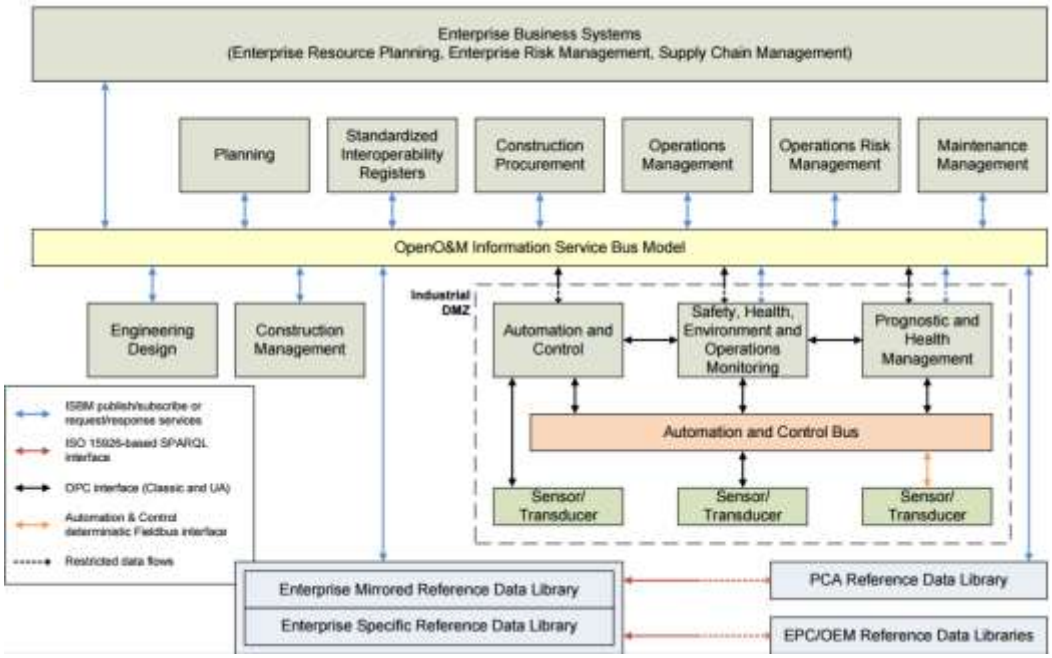

Figure 6. OIIE architecture model (Johnston, Hoppe & Sandmark 2015)

To enable all the standards to work together OpenO&M Initiative has developed system-of-systems interoperability architecture called Open Industrial Interoperability Ecosystem (OIIE). The OpenO&M Initiative was formed by ISA, MESA International, MIMOSA and OPC Foundation to name a few. Also Fiatech, POSC Caesar Association and Professional Petroleum Data Management Association have joined the work. (Mitchell 2016) OIIE defines an architectural framework for enterprise architecture. The main idea is to use the best standard for each different task and allow them to function together. OIIE has a portfolio of standards to choose the most suitable from. The architecture model has an information message bus as a “transporter”. Information models and message models are used to represent data. (MIMOSA 2016) The information message bus can be seen in Figure 6, which presents the OIIE architecture model.

To prove the OIIE concept, a public test-bed, Oil & Gas Interoperability (OGI) Pilot, is run. The purpose is also to test if the standards are actually applicable to real processes. The pilot is a debutanizer project that demonstrates the feasibility of the ecosystem from design to operation and maintenance. (Mitchell 2016) The pilot had three different companies doing the process engineering. Worley Parson produced intelligent process and instrumentation diagrams (P&ID) with XMpLant-technology. AVEVA produced the same thing using Proteus (an XML scheme) and Bentley produced Ontology Web Language (OWL) (part of ISO 15926 specification) and ecXML files. All of these were transformed into ISO 15926 -model with a transform engine created in the University of South-Australia. The ISO 15926 is again transformed to MIMOSA model. The data is then exported to CCOM-XML. The communications are done using Information Service Bus Model (ISBM). The data is then mapped to Assetricity Integrated Operations and Maintenance for Oil & Gas (IOMOG) –register. It stores all the data and takes care of mapping all the synonyms of the same piece of equipment into one. From IOMOG the data is loaded to IBM Integrated Information Core server which transforms it to ISA models and accessible with OPC/OPC UA standards. The data can be then connected to systems like OSIsoft's PI System for collecting real-time data. (Johnston et al. 2012) The whole process is summarized in Figure 7. The pilot still continues and phase two will for example add more process diagram and automation suppliers. In addition to this downstream pilot, there is also going to be an upstream pilot from the same group. (Johnston 2013)

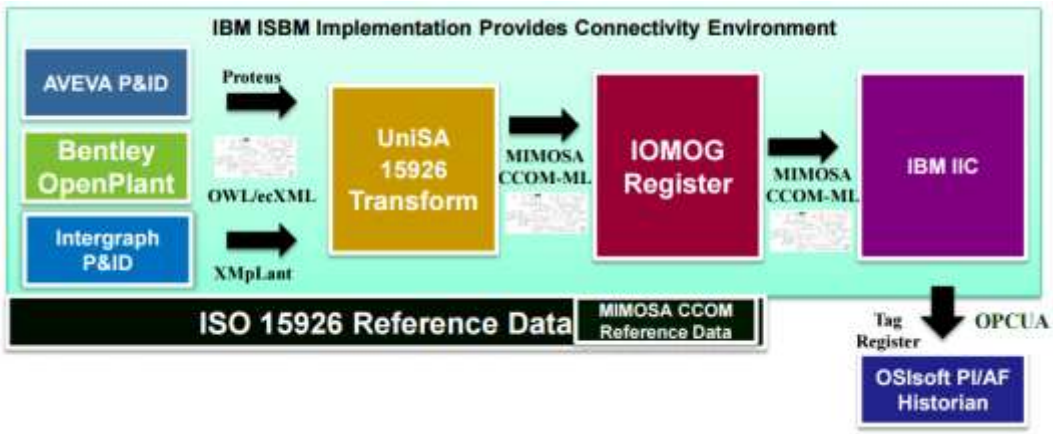

Figure 7. Oil and Gas Interoperability Pilot workflow (Johnston 2013)

3 INFORMATION MODELLING

In this chapter the concept of information modelling is introduced. First the meaning of middleware and service-oriented architecture is explained because information modelling is closely related to them. After that the definition of an information model is given together with requirements and stakeholders for the models. A wide selection of information model specifications is presented. The models discussed are OPC UA, ISA-95, ISO 15926, CAEX, AutomationML, PandIX, Common Information Model and IEC 61850. A comparison of the presented specifications is given at the end of the chapter.

3.1 Middleware and Service-Oriented Architecture

The common approach to automation system integration usually separates the automation system into layers. The structure is called the automation pyramid. The problem with this is that the information has to pass through all the layers and often there is a need for data transitions between the layers. The SOA based middleware provides a suitable way to integrate the engineering software with the process control. The difference of these two approaches is presented in Figure 8. (Melik-Merkumians et al. 2012)

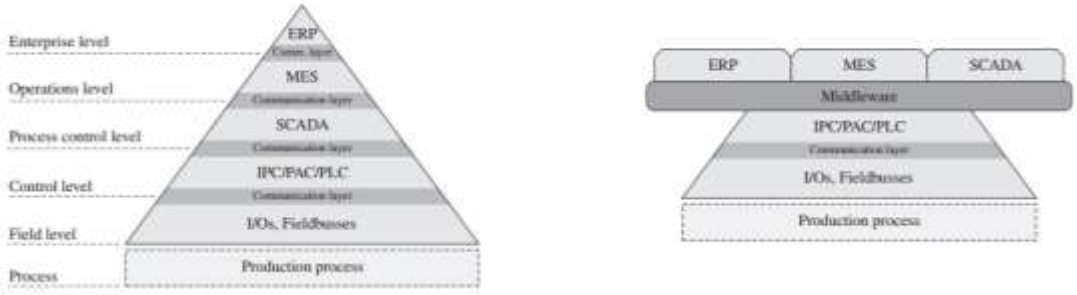

Figure 8. The differences of traditional layered automation pyramid (right) and the SOA-based approach (left) (Melik-Merkumians et al. 2012)

Middleware is not used only for exchanging data but also to provide information for example about the status of the device or the measurement range. The middleware should provide this information in a structured way and this is what requires the middleware to provide a way for information modelling. (Mahnke et al. 2011) There is a variety of different information modelling methods which are addressed in this chapter.

Another option to achieve interoperability is to implement the piece of software providing the communication functions in all the different applications of the system. Using middleware, however, benefits in reuse, distributed development, simplicity, flexibility and better maintainability. The drawback of the middleware might be defining the interface and the semantics of the information too strictly. A fixed definition is easy to start with, but not capable of accommodating to change. The middleware should provide flexible interfaces and describe the semantics in an abstract way. (Mahnke et al. 2011)

3.2 Information models

By definition information models are descriptions of certain concepts like buildings or processes. They provide a framework for presenting objects and their relationships, variables, constraints and functions. In a way, information models are a common language between software systems and devices.

There are four main stakeholders for developing communication standards and information models. First, hardware vendors want to optimize the communications of their devices. For software developers the value of the information models is in making the development and maintenance of software easier. System integrators can more easily integrate solutions of different brands and vendors. The end users achieve broader possibilities to the choice of hardware and software. (van der Linden, Granzer & Kastner 2011)

To achieve the optimal communication, there are requirements that the information models should fulfil. The most important requirement and the actual drive for developing information models further is to obtain a single comprehensive standard to describe all the equipment instead of specific description formats. Therefore the models have to be rather abstract. (Mahnke et

al. 2011) The information models have to cover at least most if not all the information required or produced within the engineering process of production systems. They have to be extendable and flexible enough to accommodate to change. The extensions could be for example vendor specific data. The representation of the data has to be efficient. Also human readable format is desired. (Schmidt, Lüder 2015)

3.2.1 OPC UA

The OPC Unified Architecture (UA) was released in 2008 by the OPC Foundation and expands the classic OPC. The classic OPC consists of many specifications, the most important being Data Access, Alarm & Events and Historical Data Access. These specifications define the access to current process data, interface for event-based information and functions to access historical data. OPC's information exchange is using client-server approach. The interfaces of OPC are based on Microsoft Component Object Model (COM) and Distributed COM (DCOM). These were used to reduce the development time and specification work. Resulting from the use of these technologies, OPC is tied to Windows operating systems. It is also one of its biggest disadvantages. (Mahnke, Leitner & Damm 2009)

OPC UA overcomes OPCs flaws by for example, being more secure and platform independent. The basic layers of OPC UA are presented in Figure 9. The base components of OPC UA are transport mechanisms and data modelling. The specification of OPC UA has 13 parts but the one addressed in this thesis is Part 5: Information Model. The whole concept of OPC UA is presented shortly before going to information models. (OPC Foundation 2015c, OPC Foundation 2016c)

The communication model of OPC UA is abstract and does not depend on protocol mappings. Currently there are two mappings, UA Web Services and UA Native. The UA Web Services mapping uses protocols like SOAP/HTTP while the UA Native uses TCP protocol. These transport mechanisms use the message-based security model from Web Services. Data modelling defines rules and building blocks to describe OPC UA information models. The access points to address space and the base types of type hierarchy are also defined. The base defined in data modelling layer can be extended to build information models.

OPC UA is based on SOA. The services are abstract descriptions and therefore protocol independent. The services provide the interface between servers as supplier of an information model and clients as consumers of the information model. (Mahnke, Leitner & Damm 2009)

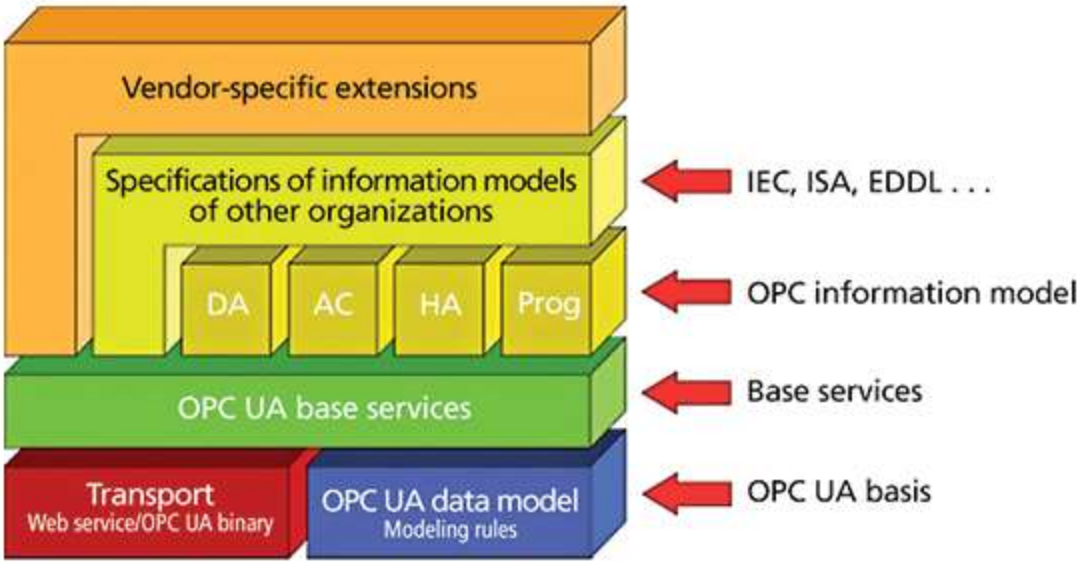

Figure 9. OPC UA information model layers. (Burke 2013)

The information modelling in OPC UA is based on nodes and references between them. The nodes can have attributes that further define them. Figure 10 gives an example of the usage of nodes. Nodes are divided to NodeClasses that include object, variable and method nodes. Variable nodes contain values with data types. They can present a value of a measurement for example. The concept of method is the same as in object-oriented programming. Method can be called with possible input arguments and it returns a result. Objects structure the Address Space. They can be used to group variables, methods or other objects by using references. This way the variables and methods belong to objects. The attributes of nodes depend on the class. For example the Variable has “Value” as one attribute. There are several attributes common for all the nodes, but the most important is the NodeId that is a unique identifier used to reference the nodes. The references are relations between two nodes. They have information about the direction of the relations, the type of the reference and of course the ids of the

source and target nodes. The reference types in OPC UA are for example HasSubtype or HasTypeDefinition. (Mahnke, Leitner & Damm 2009)

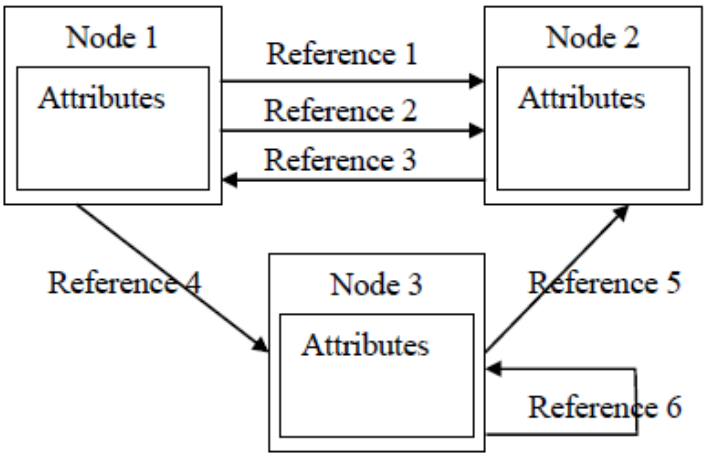

Figure 10. OPC UA Nodes, Attributes and References (Mahnke, Leitner et al. 2009)

In OPC UA information models, types are used for defining objects and variables. Variables have data types like string but also objects have type definitions that specify the type of device that the object is describing. As explained earlier, also the references have types that define them. The types can be simple or complex. Complex types can have for example references to variables and methods. Simple types just define semantics. (Mahnke, Leitner & Damm 2009) There are a lot of predefined general types in the specification. Types can inherit other types. The new derived types will have the same properties as the "parent" type but can also have own extended properties. For example, if there is a type called VesselType and it has a reference to a variable called Diameter, its subtype, let's say PressureVesselType will have the same referenced variable. In addition the PressureVesselType could have a variable called MaxPressure. All the object types are inherited from BaseObjectType, all the variables from BaseVariableType and so on also for reference, data types and events. (OPC Foundation 2015b)

OPC UA has several companion specifications. They include specifications for Analyzer Device Integration (ADI), PLCOpen, Field Device Integration (FDI),

Device Integration (DI), ISA-95, AutomationML and AutoID. (OPC Foundation 2016d) The hierarchy of the information models was presented in Figure 9.

3.2.2 ISA-95

ISA-95 is developed by the International Society of Automation. It contains models and terminology to define a format for information exchange between different systems. The ISA-95 standard has five parts. In the scope of this thesis is Part 2 Object Model Attributes, but also a brief overview is given. (ISA-95 2015)

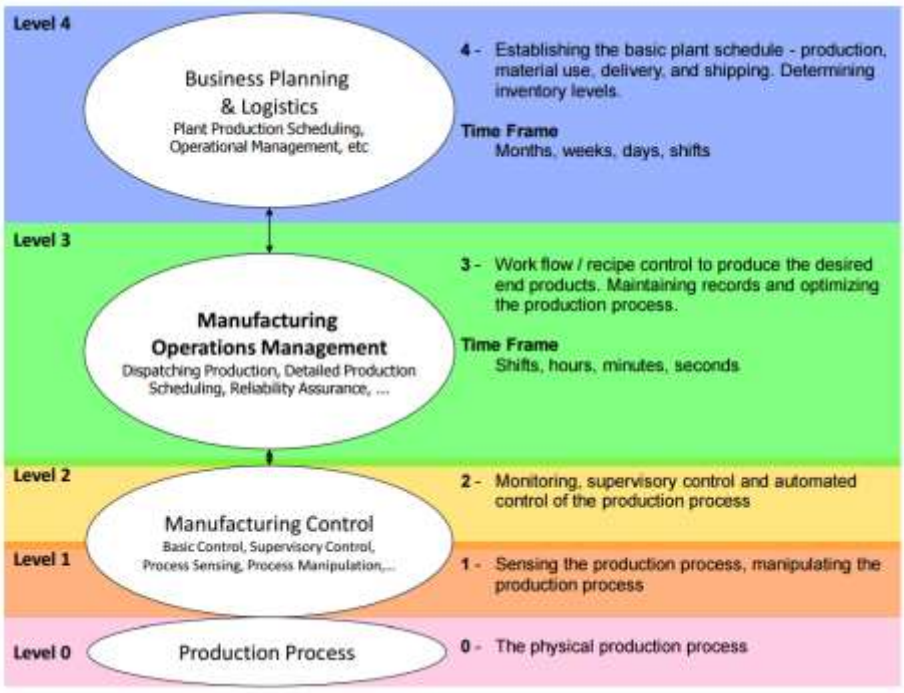

Figure 11. Activity levels of ISA-95 (OPC Foundation 2013b)

ISA-95 standard defines five activity levels for a manufacturing organization. Level 0 defines the actual manufacturing process while levels 1 and 2 are the automation and control. Level 3 is manufacturing operations management (MOM) level, containing for example MES applications and level 4 is business planning and logistics, meaning ERP for example. The levels of ISA-95 are presented in

Figure 11. The main focus in the standard is the information exchange between levels 3 and 4 and across level 3 systems. (OPC Foundation 2013b)

The part 2 of ISA-95 lists industry-independent information as attributes that can be used to define processes. Industry specific and application specific information is characterized as property objects. These industry-independent attributes include the resource models. They are Personnel, Material, Equipment, Physical Assets and Process Segments. (OPC Foundation 2013b) These models and the overview of the ISA-95 are presented in Figure 12.

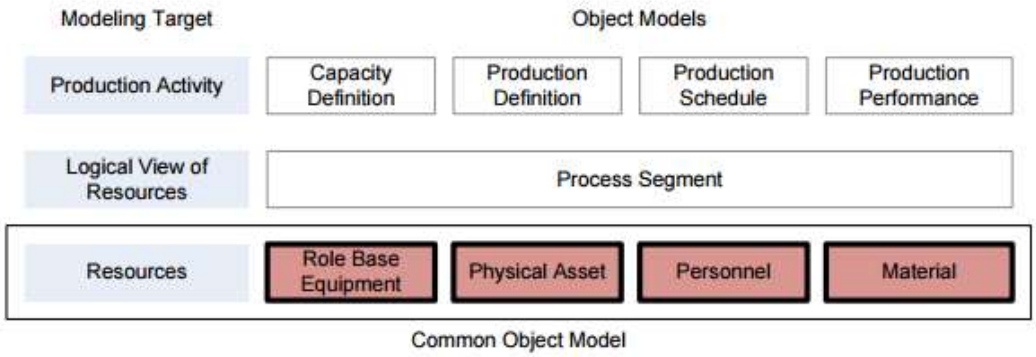

Figure 12. ISA-95 Overview. (OPC Foundation 2013b)

For representing information as objects ISA-95 uses Unified Modelling Language (UML). A set of attributes is associated to these object models. A UML presentation of the Equipment model is presented in Figure 13. It includes the definition of “Equipment” and “Equipment Class” which are the definitions of the equipment type in the production, for example the class could be a tank. The classes also have properties. (OPC Foundation 2013b)

ISA-95 is an abstract specification since it doesn’t provide implementation. There are some implementations such as Business to Manufacturing Markup Language by MESA. (OPC Foundation 2013b)

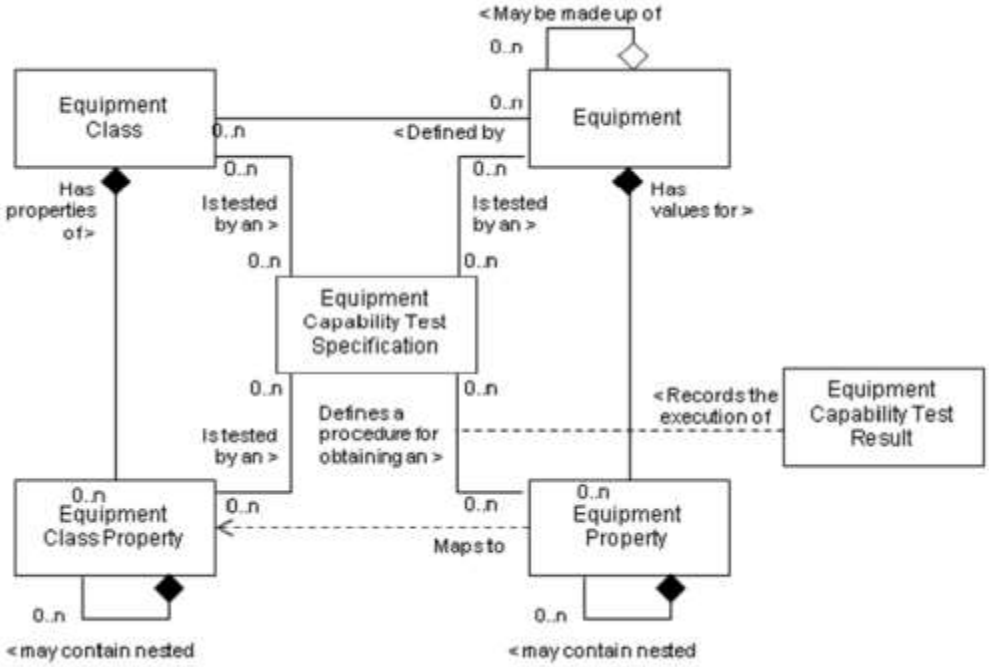

Figure 13. ISA-95 Equipment Model (OPC Foundation 2013b)

3.2.3 ISO 15926

ISO 15926 (Licycle Information Exchange) is a standard for representing information related to engineering, construction and operation of a process plant. It tries to cover the whole life-cycle of a plant. It is specially meant for the O&G, but since it is generic model it is applicable for other industries as well. (ISO 15926-1 2004) ISO 15926 has currently 8 parts and two parts are still under development. (ISO/TS 15926-11 2015)

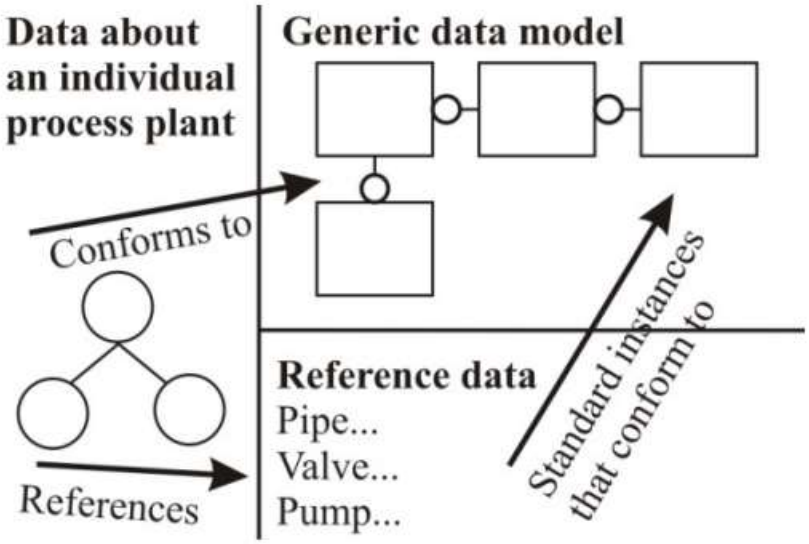

Figure 14. ISO 15926 architecture (ISO 15926-1 2004)

The second part of ISO 15926 defines a generic data model that is used to represent and exchange the life-cycle data. It establishes the basic entity types and their connections. The entity types are not enough to represent a plant and detailed information about the objects, like pipes, needs to be added as reference data. The reference data consists of classes that define plant objects. It is organized in the reference data library which is accessible through the reference data services. The reference data is standardized in parts three and four of ISO 15926 but additional reference data can be created by authorized users. (Holm et al. 2012) The ISO 15926 architecture is presented in Figure 14.

The entity types of ISO 15926 can be hierarchically ordered using subtype and supertype relationships. Another major modelling strategy is temporal and special composition by relation entities. In the hierarchical model subtypes are derived from root element “thing”. The root element carries information about identity and derived abstract basic entities class, relationship and multidimensional object. This is presented in Figure 15. Attributes are implemented as instances of basic data types or references to other elements. (Mahnke et al. 2011)

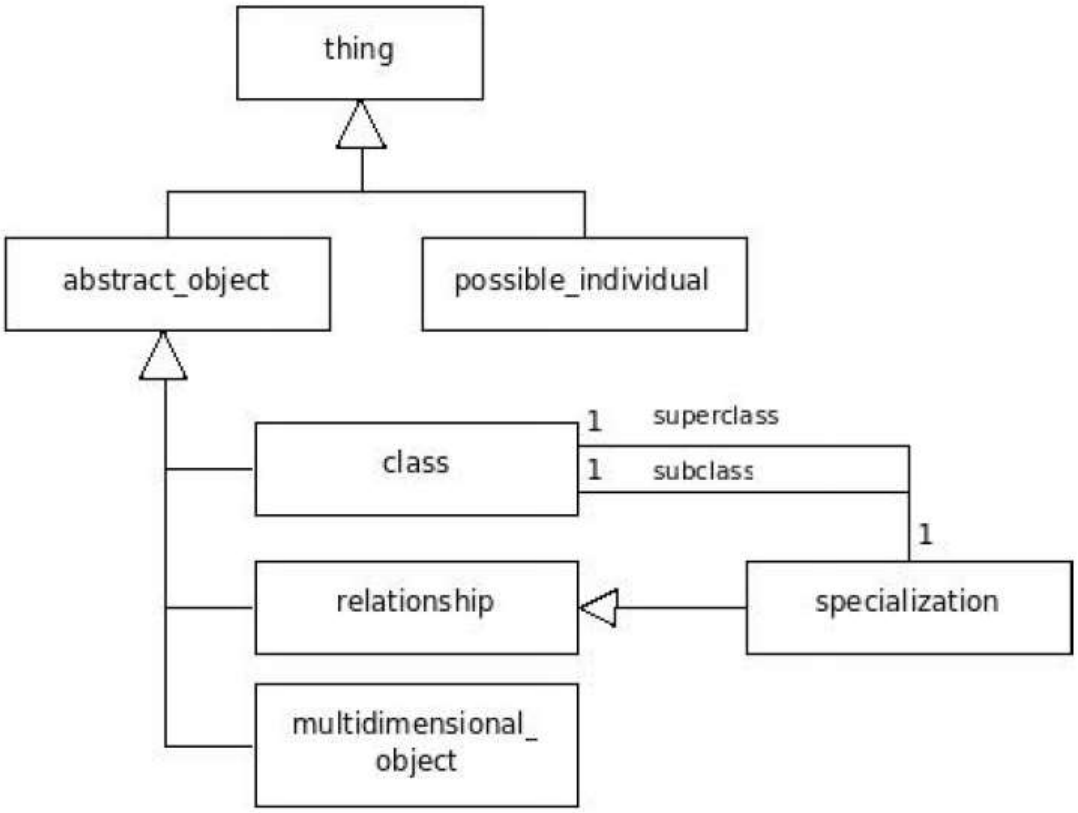

Figure 15. The basic model elements of ISO 15926 (Mahnke et al. 2011)

3.2.4 CAEX

Computer Aided Engineering Exchange (CAEX), or also known as IEC/EN 62424, is a data format that provides meta-model that can define hierarchical plant models and define attributes for models. It defines elements, interfaces and components and concepts for modelling relationships and functional and topological hierarchies of them. These concepts can be used as a model for information exchange between engineering software tools. (Mahnke et al. 2011) Especially CAEX is focused on the exchange P&I diagrams from tools that create them to process control engineering or computer-aided engineering tools. (Holm et al. 2012)

CAEX defines three class libraries which are SystemUnitClassLib, RoleClassLib and InterfaceClassLib. The InterfaceClassLib contains interfaces for modelling the information flow between resources and controls systems or mechanical connections like flange for connecting pipes. A class in this library defines type of the link between elements. It can also have attributes, like “direction”. The RoleClassLib comprises RoleClasses that are used to model functions of objects. The functions are something that the technical implementation has to fulfill, like for example "a conveyor". The roles are also used to assign graphical images to the objects. An object with the role "conveyor" would have a corresponding image in the diagram. The SystemUnitClassLib holds logical and physical plant objects. They are like classes of object-oriented programming. They describe the system elements in detail by defining the content and meaning of the elements. Roles can be assigned to these classes and with roles the elements get new attributes and interfaces. The classes from SystemUnitClassLib are used for instantiating InternalElements which are instances of these objects. The plant hierarchy is constructed in SystemHierarchy, which is kind of a container-object of the model. (Schleipen 2010, Holm et al. 2012) Figure 16 shows how all these libraries are used together.

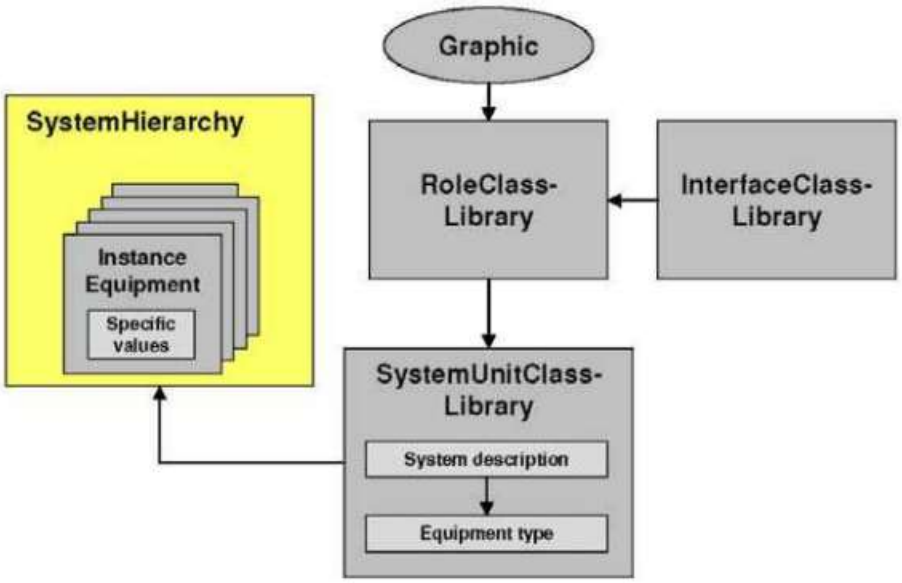

Figure 16. The usage of CAEX libraries. (Schleipen 2010)

3.2.5 AutomationML

AutomationML (Automation Markup Language, AML) is a data format developed by Daimler, ABB, Siemens, Rockwell, Kuka, Zühlke, netAllied and the universities of Magdeburg and Karlsruhe. The idea behind AML is to be a neutral format that serves for data exchange between manufacturing engineering tools, like CAD or simulation tools. (Drath et al. 2008)

Like many other standards, AML is object-oriented and describes plant components as data objects. The objects can consist of sub-objects and be part of some higher level object themselves. Data objects can be everything from robots or signals and values to tanks and manufacturing cells. The aspects depicted are for example objects position in plant topology, relations to other objects, its behavior, kinematics or geometry. (AutomationML consortium 2010)

AML uses established data formats for different aspects. It mainly serves as integration format between the standards and defines how to use them to achieve interoperability. The standards useds are CAEX, PLCopen XML and COLLADA™. (Drath et al. 2008)

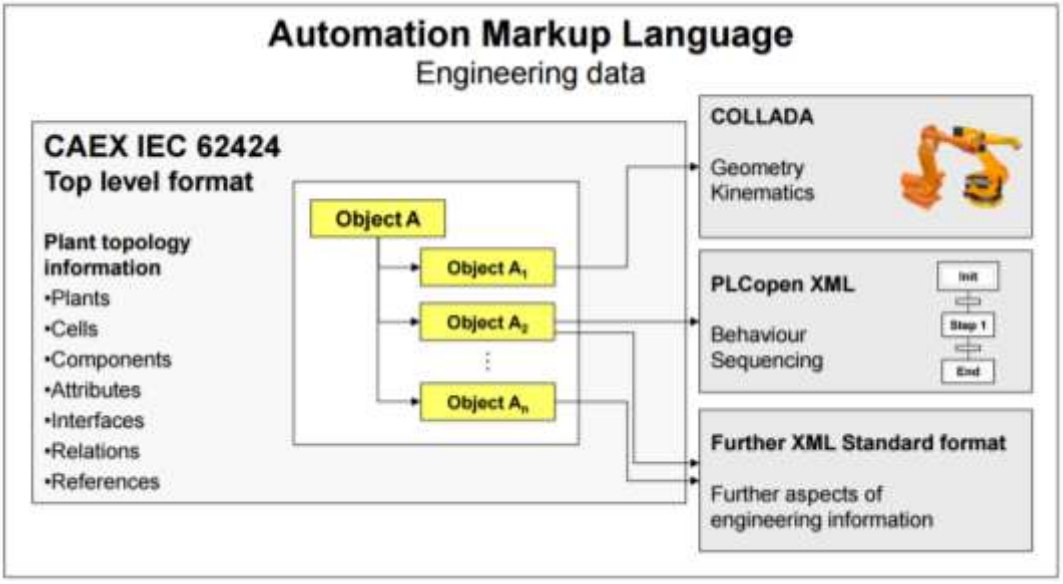

Figure 17. Basic architecture of AutomationML. (AutomationML consortium 2010)

For plant topology, AML uses CAEX. Because topology is also the top level of AML, CAEX is the high level integration frame of it. A specific usage for the format is defined. For geometry objects COLLADA™ -format is used to store them in separate XML files. Also kinematic information is stored similarly. The logic information is described with PLCopen XML –format as sequential functional charts (SFC). The variables in SFCs can be published as CAEX ExternalInterfaces so that the high level interconnections can be presented. The links between CAEX and external files, references, and the connections between CAEX objects, relations, are presented with standard CAEX mechanisms. The use of these standards and the basic architecture of AML are presented in Figure 17. There are several advantages from the AML architecture. When established data formats are reused, the specification effort of AML is reduces. The data is distributed into different files and the bulk data handling is therefore easier. The library files usage is simplified by storing and exchanging them separately. The geometry and logic variants can be stored separately to distinguish between degrees of detail. (Drath et al. 2008)

For the application of CAEX, AML defines certain rules and special libraries. AML defines how to identify objects and classes. InterfaceClassLib contains several interface classes for general automation systems. The classes of the library allow modelling of user defined interface instances. RoleClassLib defines the role classes that explain the functionality of CAEX objects. AML standard doesn't define specific SystemUnitClassLib, but it does define some rules for it. InstanceHierarchy stores project data and is the core of AML data. It is hierarchy of object instance and its properties, references and relations. (Drath et al. 2008)

3.2.6 PandIX

PandIX is an information modelling method developed to exchange the data of P&I diagrams. It describes the functionality of the plant structure for control purposes in a standardized way. It doesn't model any other relations, like chemical or physical reactions or balances. PandIX extends the CAEX model and it provides interfaces for interoperability with CAEX. Also interfaces are provided for vendor-specific solutions and XMpLant which is based on ISO 15926. PandIX was developed in Aachen University. (Schuller, Epple 2012)

PandIX uses interface based on CAEX to export the information related to functionality of the plant from the engineering software. The information that is irrelevant for process control engineering is not exchanged. The PandIX model is specified as a meta-model. The specification contains the model description, a library of standardized process plant elements, a suggestion for positioning system and mapping rules to create CAEX XML file for the model. (Schuller, Epple 2012)

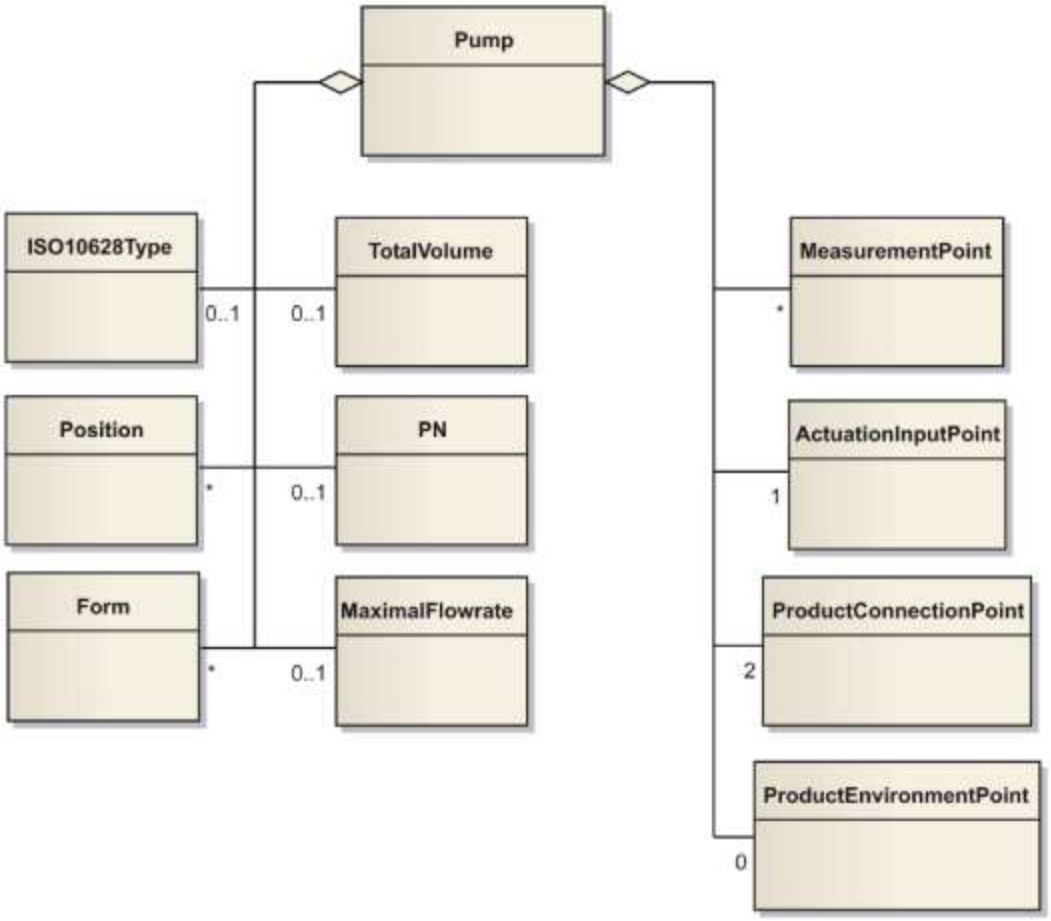

Figure 18. Example of process plant element in PandIX (Schuller, Epple 2012)

PandIX model has two types of technical units. They are process plant elements and process control elements. The plant units are for example pipes and vessels

and the control units are for example sensors. Both element types must have at least one interface. Links are used to connect interfaces. These links cannot have any functional properties since every functional connector, like pipe, is modelled as an element. In addition to having interfaces process plant elements have always at least one channel for products. The channels also have interfaces. A channel describes the flow of the actual product inside the process element. A pipe would have one channel for example while a heat exchanger would have two. An example of a process plant element is presented in Figure 18. The variables of the element are on the left side and the interfaces on the right. Process control elements are used to send and receive information between the real and the virtual world. They have interfaces for signals and control. The signal interface enables connecting two control elements and the control interface enables transferring information to process plant element. (Schuller, Epple 2012)

PandIX also provides form and positioning information. These can be used for example to describe geometry of vessels or positioning of a sensor. There are two ways to define this kind of information in PandIX. Other one is to export a complete 3D model, and the other is to add only the necessary pieces of information to PandIX. (Schuller, Epple 2012)

3.2.7 IEC 61970/61968 Common Information Model

The IEC 61970 is a model used to define the components of a power system and their relationships at an electrical level. IEC 61968 goes hand in hand with IEC 61970 since it extends the model to cover also the other aspects of power system software data exchange. These could be billing, asset tracking and work scheduling for example. The standards also define Common Information Model (CIM) for power systems. The primary use of the standards is to facilitate the exchange of power system network data between companies and to allow the exchange of data between applications. (McMorran 2007)

CIM is described using UML concepts. The IEC 61970-301 specifies the core packages and the IEC 61968-11 brings additional packages. The CIM consists of multiple main packages with different functionalities. There are also sub packages and classes with attributes and associations. Physical objects like equipment and abstract objects like operations can be described with this set of

abstract classes, attributes and associations. (Rohjans, Uslar & Appelrath 2010) The basic concept of CIM is presented in Figure 19. As can be seen classes can have subclasses and relationships to other classes. They have also attributes. Classes belong to packages which can be nested.

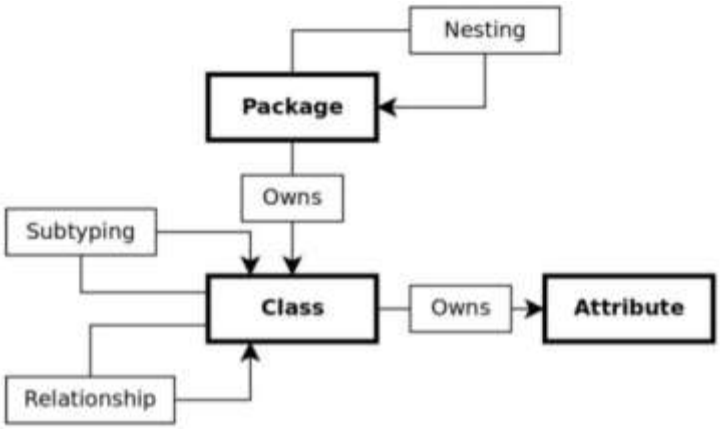

Figure 19. Basic concept of CIM. (Mahnke et al. 2011)

An example of the usage of CIM is given in Figure 20. It is a model of a steam turbine. Inheritance is presented with arrows. For example, StreamTurbine-object inherits PrimeMover-object. There are also references. The StreamTurbine has SteamSupplys for example. Almost all the objects have attributes. The basic attributes required for identifying an instance of an object can be seen in Core::IdentifiedObject.

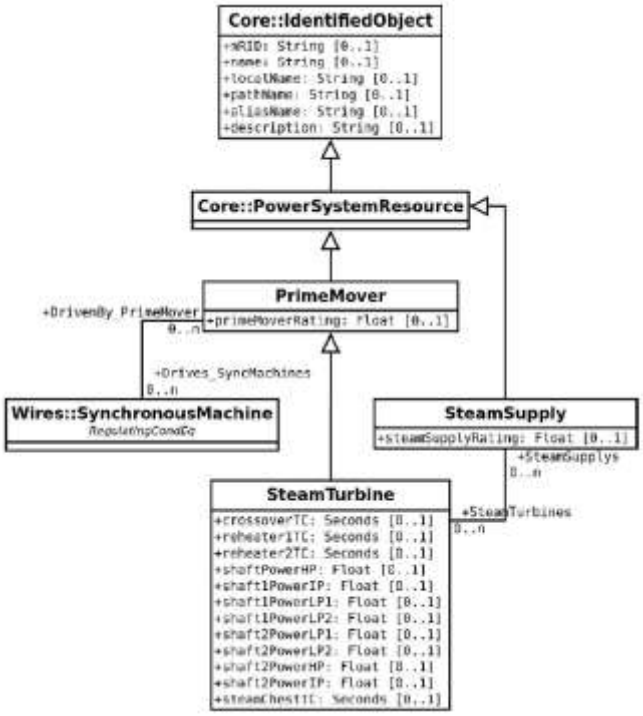

Figure 20. Steam turbine CIM model. (Rohjans, Uslar & Appelrath 2010)

3.2.8 IEC 61850

As the name of IEC 61850 tells, it is a standard for “Communication Networks and Systems in Substations”. Although IEC 61850 is mainly a communication standard, in Part 7-4 it also defines basic information model used to describe specific substations, hydro-power generation and decentralized power generation. IEC 61400 extends this list with wind power generation. (Kostic, Preiss & Frei 2003, Mahnke et al. 2011)

The modelling concept of IEC 61850 is object-oriented. It supports objects, attributes, data types and aggregation. Inheritance is not supported. (Mahnke et al. 2011) In the model, the main abstraction type is a logical node. The nodes can represent two things. Either they depict a function of substation the automation system or they depict external process equipment. The nodes contain a hierarchy

of data objects and the objects contain attributes. The attributes store process information as well as configuration information etc. (Brunner 2008) In addition each attribute has a type, like boolean or integer. The information model can be seen in Figure 21. The standard defines several domain specific logical nodes. To extend the model, logical nodes and object classes can be added by the user. The information model is designed with and information exchange model. The standard specifies also a Substation Configuration Language (SCL). It can be used to exchange configuration information between tools. (Mahnke et al. 2011)

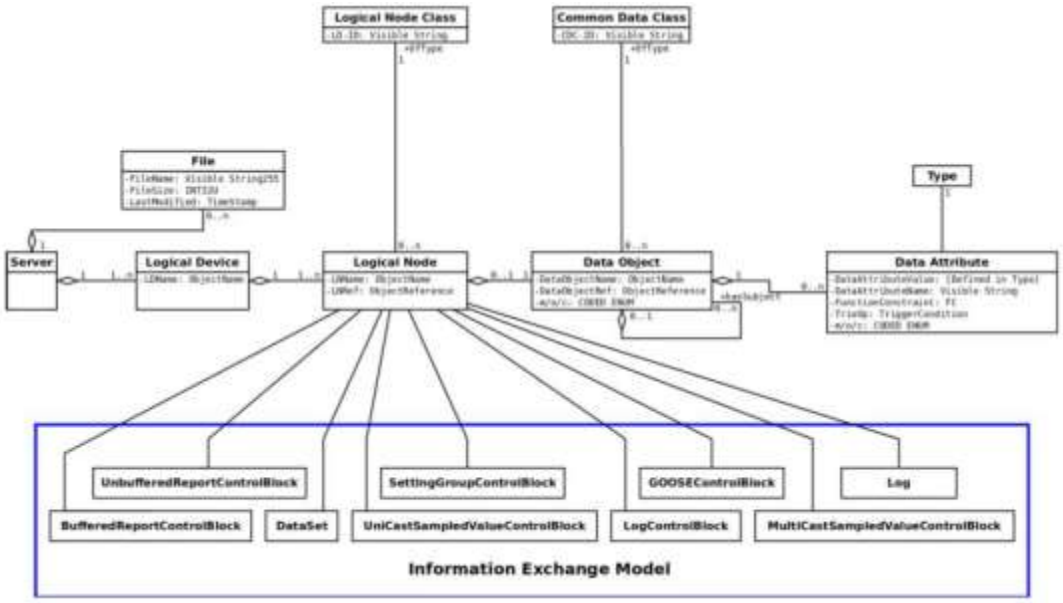

Figure 21. The IEC 61850 Information Model (Mahnke et al. 2011)

3.3 Comparison of standards

To obtain understanding about the modelling capabilities of the standards a simple comparison was made. The necessary requirements for all the information models are hierarchy, aggregation and variables. They are needed to present the structure of the process or the plant. Some of the models don't support functions or concepts of object object-oriented programming. However, these capabilities

are not needed for the uses of the models lacking them. (Mahnke et al. 2011) The results of the comparison are presented in Table 1. The comparison is similar to the one made by Mahnke et al. (Mahnke et al. 2011), but AutomationML and PandIX were added to it. (Schmidt, Lüder 2015, Schuller, Epple 2012) Also the support of inheritance and classes was added to CAEX according to Schleipen (Schleipen 2010). As can be seen from the table OPC UA is the standard with the strongest information modelling capabilities.

Table 1. Comparison of the information modelling standards (Mahnke et al. 2011, Schleipen 2010, Schmidt, Lüder 2015, Schuller, Epple 2012)

	OPC UA	ISA-95	ISO 15926	CAEX	AML	PandIX	IEC 61970 (CIM)	IEC 61850
Hierarchy	x	x	x	x	x	x	X	x
Aggregation	x	x	x	x	x	x	X	x
Variables	x	x	x	x	x	x	X	x
Functions	x	x	-	-	-	-	-	-
References	x	x	x	x	x	x	X	-
Classes	x	x	x	x	x	x	X	Partly
Methods	x	-	-	-	-	-	X	-
Inheritance	x	x	x	x	x	x	X	-
Data Types	x	-	x	x	x	x	X	x

Other important concept of information modelling is, as mentioned earlier, the extensibility or flexibility of the information models. This is needed to add vendor or end-user specific data. The concreteness of the model is also significant to be able to understand how domain-specific the models are. The models do also have different philosophies on how to model the system. The comparison of these features is presented in Table 2. (Mahnke et al. 2011)

It is clear that there are lots of standards with different use cases and purposes. At the moment it is impossible to find a single standard applicable for all the uses. This leads to heterogeneous system of software build on different standards. (Selway et al. 2015) Therefore one significant factor in choosing the standard is how compatible it is with other standards. The best selection of the standard should fit to the use case, in this case process industry and specifically downstream O&G. The standard chosen should also be fairly popular to be supported by many devices and software systems.

OPC UA as well as ISA-95 is developed to be used in all kinds of industries and everything from batch processes to continuous processes. (Mahnke, Leitner & Damm 2009, OPC Foundation 2013b) ISO 15926 on the other hand has a clear scope in upstream O&G, but being a generic model, it is applicable to other processes as well. (Holm et al. 2012) CAEX aims to be a standard for the exchange of data between P&I diagram development tools and process control engineering tools. (Holm et al. 2012) AutomationML is a standard developed for production systems engineering and commissioning. (Schmidt, Lüder 2015) PandIX is meant for the exchange of P&I diagrams. (Schuller, Epple 2012) IEC 61970 and IEC 61850 are related to energy generation and grids. IEC 61970 models components of power systems and their relationships. (McMorran 2007) IEC 61850 is for describing substations, hydro-power generation and decentralized power generation. (Mahnke et al. 2011)

Table 2. Comparison of the qualitative attributes of the standards.

	Extensibility	Philosophy	Concreteness	Source
OPC UA	Extensible in several ways, adding type hierarchies and reference types	Object-oriented	+	(Mahnke, Leitner & Damm 2009)
ISA-95	User specific data types and property values	Object-oriented	+	(Mahnke et al. 2011)
ISO 15926	Authorized users can add reference data	Ontology	++	(Holm et al. 2012)
CAEX	External interfaces	Meta	--	(Holm et al. 2012)
AutomationML	User defined classes and libraries	Meta	-	(Schmidt, Lüder 2015)
PandIX	External interfaces	Meta	-	(Schuller, Epple 2012)
IEC 61970 (CIM)	Extensible classes, attributes, pakcages, methods	Object-oriented	++	(Mahnke et al. 2011)
IEC 61850	Extensible logical nodes and objects	Object-oriented	++	(Mahnke et al. 2011)

OPC UA seems to be the standard to which other standards are relying on and trying to find compatibility. ISA-95 and AutomationML for example have OPC UA specifications. (OPC Foundation 2016d) According to Mahnke, Gössling and Graube, OPC UA can be used to map CAEX, ISA-95, ISO 15926, IEC 16970 and IEC 61850 information models. (Mahnke et al. 2011) Most likely also PandIX is compatible with OPC UA, since it is also based on CAEX.

Because of the above mentioned reasons, OPC UA is selected to be the standard used for information modelling in this thesis. The advantages in selecting OPC UA are that OPC UA is highly platform independent standard. It supports complex data types and object models. It is capable of achieving high speed transfers by using efficient binary protocols. One of the main reasons for selecting OPC UA is that OPC UA has a broad industry support and it is being used to support also other standards like ISA-95, ISA-88, EDDL and MIMOSA. (Postół 2015) It is supported also by Germany's Industrie 4.0. (OPC Foundation 2015a) The drawback of OPC UA is that it isn't capable of addressing the whole life-cycle. It is focused only on the operational phase. (Mahnke et al. 2011)

4 DEVELOPMENT OF INFORMATION MODELS

This chapter explains the process of designing information models. In the previous chapter, different information model specifications were introduced and compared. According to the comparison made OPC UA provides the widest information modelling capabilities and was selected to be used in this thesis. In the first subchapter a rather generic modelling process is explained. The second subchapter gives some rules regarding information modelling in OPC UA. The existing modelling tools are reviewed in the last subchapter.

4.1 Information modelling process

Harju (Harju 2015) proposed steps for designing information models in his thesis. At first, the designer should get to know the process and the equipment to be modelled. That is done in three steps. First, data is gathered from P&IDs, process experts and other sources. At the same time the equipment related to the process are discovered. Next the signals such as measurements available from the process are studied to figure out which of them are necessary for the model. The gathered data needs to be validated before continuing from this initial step forward. (Harju 2015) In addition to the information of the process and equipment, also defining the requirements is important. The requirements help to understand the needed level of detail of the model and what information should be focused on. This kind of information can be gathered from the stakeholders of the information models by asking what is going to be done with the model and how. They could be for example clients or the ones maintaining the address space. The defined requirements should also be validated.

After familiarizing the process, the modelling tools can be selected. Of course there is always the option to create models by writing with a simple text editor. However, tools can be helpful for modelling and maintaining the server address space. Three existing modelling tools are presented briefly later in the second subchapter. Creating own tool might be a feasible choice for maintaining the models. (Harju 2015) After the tools are set up and all the information is gathered, the actual modelling can be done.

First the devices are modelled. The existing types can be inherited and expanded to match the devices. The modelling should start from the highest level of abstraction and move towards smaller components. For example if a whole plant is modelled, the plant is the highest level but if a simple process is modelled, the process unit should be the highest level. From process unit the next level could be devices, then analyzer devices or sensors and finally simple input signals. Subchapter 4.3 gives some rules for information modelling with OPC UA.

When the devices are modelled and the equipment hierarchy is ready the model can be moved to OPC UA server. Many of the information modelling tools provide a code generation function. The code can be inspected and own changes can be made before the server is set up. After the server is running instances can be created to server address space and used. (Harju 2015)

4.2 Existing modelling tools

Information models can be created by writing code directly or using some graphical editor that handles the code generation. Using these modelling tools brings some benefits. First of all, the tools make information modelling possible for users that don't know how to code. Even for coders the tools can provide validation and ensure that the produced model is error free and valid OPC UA. The tested modelling tools in this thesis were OPC UA Address Space Model Designer from CAS, UaModeler from Unified Automation and OPC-UA-Modeler from Fraunhofer IOSB. The two first of these are freely available for testing, while the third one is not provided as a demo version. The selection of tools does not intend to be exhaustive.

4.2.1 OPC UA Address Space Model Designer

OPC UA Address Space Model Designer is available in two editions, professional and standard. The main difference is that the professional version is capable of importing and exporting XML schemas or UML while the standard is not. The professional version can also be used for UA Server Configuration and has some other more advanced features. The standard version can be used for basic modelling and also supports publishing the model as OPC UA Address Space. (CAS 2011a) The version tested was 3.20 Professional.

The OPC UA Address Space Model Designer user interface is divided into two panes. Figure 22 represents the user interface. On the right side, it always shows properties and data bindings of an object clicked. On the left side it has different views that are called “Model”, “Browse View” and “Model 3D”. The “Model” shows the information model currently under development and a type library. The “Browse View” shows the Address Space of the models as it can be seen on the server. The “Model 3D” is a graphical representation of the model. It seems rather complex, unclear and hard to use. In addition to these the designer tool provides help pane with lots of information about OPC UA. (CAS 2016)

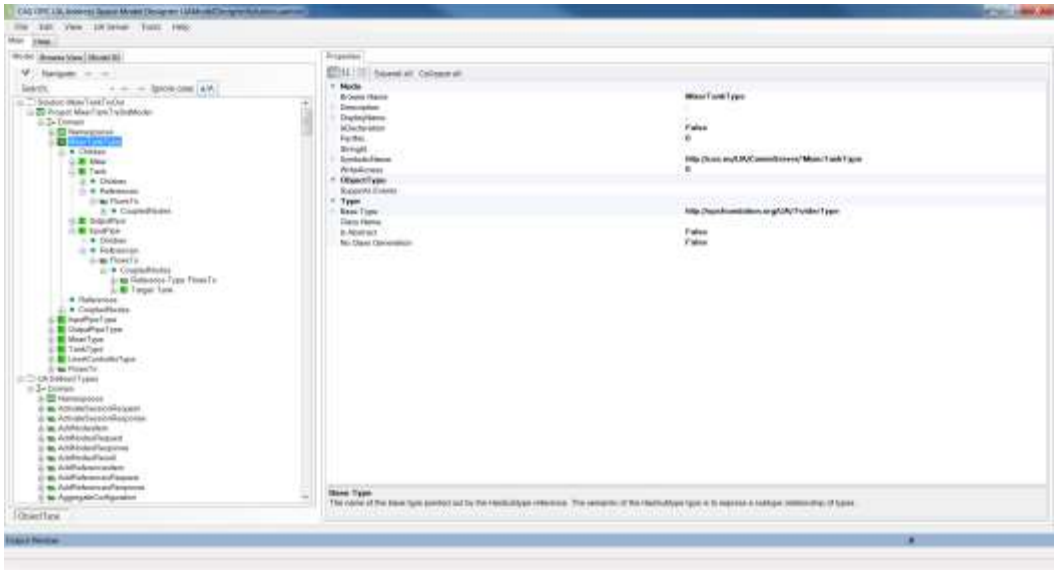

Figure 22. The user interface of OPC UA Address Space Model Designer.

The tool currently has many different features under development. Many of them are helper functions like undo and redo. One interesting feature of the tool is the data binding functionality. It allows binding process data to the model at the modelling phase. Also other plug-in tools can be added. (CAS 2016)

The OPC UA Address Space Model Designer can load models from XML-files and its own file format. Also saving is possible. Import can be done from UA node set XML-files. OPC DA server address space can be imported. Multiple import

formats, like UML diagram, XML schema and Visio are under development. The same formats are also under development for export. The model documentation can be exported to a Word document or Microsoft Assistance Markup Language (MAML). (CAS 2016) The tool is capable of exporting XML-files and generating C#-code to present the functionality of the model and to create the node instances. It must be noted that the functions created are just empty function stubs. Also other code generators can be added. (CAS 2011b)

4.2.2 UaModeler

The version of UaModeler tested for this thesis was 1.4.3. The UaModeler provides a graphical interface to build OPC UA information model. The tool has two modelling views. The one called “Graphics View” is the more graphical presentation of the model. It shows the objects and their relationships in a diagram. The other one is called “Nodes View” and provides detailed information about nodes, their attributes, children etc. The user interface and nodes view can be seen in Figure 23. For modelling purposes the node view seems more practical even though the graphical view allows seeing the relationships more clearly. In addition to these views the user has information model and project panes always at sight. At least the information model pane seems useful since it provides a clear view of the model hierarchy. (Unified Automation 2016a)

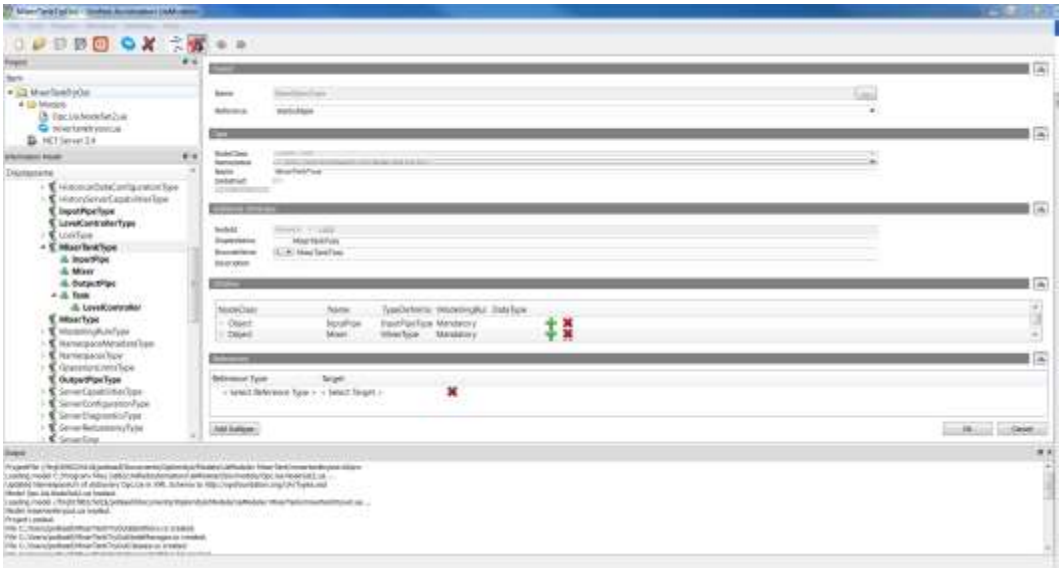

Figure 23. The user interface of UaModeler.

The UaModeler provides standard OPC UA node set and in addition to that it comes with node sets for PLCOpen and Device Integration (DI). Other models have to be added by the user. For example, the ISA-95 model can be found from the OPC Foundation web site. (Harju 2015) The models can be imported from XML files or from UaModelers own file format. XML is also used for exporting the models. For generating code UaModeler has several options. C and C++ server code as well as a .NET server or client C# codes can be generated with UaModeler. The codes are used to provide function stubs or create instances of the model nodes. XML is required for providing the structure and type definitions. (Unified Automation 2016a) Other licenses for code generation can be purchased from Unified Automation. (Unified Automation 2016b)

4.2.3 OPC-UA-Modeler

OPC-UA-Modeler from Fraunhofer IOSB provides a graphical view to develop information models. One interesting feature of the tool is that it supports importing CAEX and AutomationML formats. It can be used to set up OPC UA servers using CAEX or AML files. As well as the other tools, also OPC-UA-Modeler supports XML import and export. The application is based on Windows Presentation Foundation (WPF) and Silverlight. It is possible to get the application in German or English. (Harju 2015, Fraunhofer IOSB (c) 2016)

4.2.4 Comparison

Of the two tested tools, the UaModeler from Unified Automation has the best user interface. It is clear and easy to use. The hierarchy of the nodes is clear and the editor has excellent helper functions like automatically filling some information or an easy way to add children to a node. OPC UA Address Space Model Designer from CAS has the largest set of features and some really useful features under development. The comparison of the tools can mainly be done between these two, since the Fraunhofer OPC-UA-Modeler lacks important features such as a code generator. Fraunhofer's tool however provides CAEX support which makes it stand out a bit. In this thesis CAEX support is not important.

All the tools provide XML import and export capabilities. A graphical view is available in all the tools although the 3D view in OPC UA Address Space Model

Designer was quite hard to use and interpret. The Address Space Model Designer and UaModeler both provide code generators. UaModeler can generate C, C++ and .NET server (C#) code, while the Address Space Model Designer can create C# code. Other compilers can be added to both of tools. Both of these tools use XML for type definitions and providing the structure of the information model. The code is used for creating instances of nodes and to provide function stubs that require implementation. The binding of process data is a unique feature for Address Space Model Designer and makes it stand out. However because of the complex and less intuitive graphical user interface, it is harder to use than the UaModeler.

Even when using modelling tools, some things still need to be done by hand. Both, the UaModeler and the Address Space Model Designer are capable of creating methods. The methods are generated with input and output parameters but the actual method code has to be added by the user after the method stub has been generated by the tool.

Because of the well-designed user interface of the UaModeler it would be the first choice out of these three modelling tools. It is a lot easier to use than the OPC UA Address Space Model Designer and the models can be created with a lot less clicks. Also the validation during design is better in UaModeler. For example, if ReferenceType is removed, all the references are removed also. Address Space Model Designer validates the code when compiling. The code generated with UaModeler is more easily readable than the one from Address Space Model Designer. Although it must be said that Address Space Model Designer uses the code compiler provided by OPC Foundation. In functionality the Address Space Model Designer is as good or even better choice than the UaModeler.

4.3 Generic rules for information modelling with OPC UA

When creating information models, the model should always be kept as simple as possible for the use case. The level of detail in the model depends on the requirements and the data source. If the client only needs to access some values with ids, a full information model is unnecessary. If the data source is only a simple OPC DA server it is impossible to create a rich information model. (Mahnke, Leitner & Damm 2009)

OPC UA allows the developer to define more data types freely to match the application needs. There is also existing set of types and instances. The new data types inherit the existing type from which they derive from but they can also have modifications to the features. New data types must be exposed to the Address Space by the server. (Postół 2015) The data types used should be standardized types if possible. The standardized types can be selected from OPC UA specification or from a selected companion specification. (Mahnke, Leitner & Damm 2009)

Generally speaking there are two ways to design new information models that describe the behavior and state of a process. One is to adopt an existing model from a companion specification and the other to design a custom model with own data types. To unify the information models and to promote reusability, OPC UA has many companion specifications for different processes. (Postół 2015) The standard information models defined in the companion specifications should be used when available because they might be familiar to the clients. (Mahnke, Leitner & Damm 2009)

4.3.1 Structuring

In OPC UA objects are used to access methods and variables. The detail of structure of the model depends on the usage. The more the client needs to browse the address space the more structured it should be. Grouping of nodes and variables can be done in several ways. Nodes should match devices or similar instances. The variables under them can be grouped by objects according to the purpose of the variable. The device nodes can be ordered according to location, functionality etc. OPC UA allows having multiple hierarchies. (Mahnke, Leitner & Damm 2009)

The ReferenceTypes can be used to define relationships between the nodes. There are hierarchical and non-hierarchical references. When existing ReferenceTypes are not enough, new types can be made. A supertype should be selected carefully for a new ReferenceType. Appropriate supertypes allow easy filtering of Address Space. (Mahnke, Leitner & Damm 2009)

Views can be used to show parts of Address Space while hiding unnecessary information. They either provide a hierarchy or to hide subcomponents of a node. (Mahnke, Leitner & Damm 2009)

In addition to structuring a single information model, also the whole namespace needs some kind of form. To keep the structure of the models easily understandable, all the different plants types should have their own namespaces. (Harju 2015) For example a distillation unit and cracking unit should be in their own separate namespaces.

4.3.2 Defining types

All the objects and variables need to have an OPC UA type definition. If server is lacking types, BaseObjectType and BaseDataVariableType can be used. PropertyType is always used for properties. The definitions give more information about the objects. Therefore providing specific type definitions is useful and especially so when the definition is specified in a standard information model. This is because the client applications also use those type definitions. (Mahnke, Leitner & Damm 2009)

Before creating new type definitions some things should be considered. If it is possible to provide the same information with a standard type definition, a new one shouldn't be created. Subtypes or instance specific information can be used to extend those standard definitions. If the decision is made to create a new type definition, a suitable supertype needs to be selected. There are always the base types but if a more specialized supertype is available, it should be used. The supertype is always specialized more by the subtype by for example adding semantic. A supertype cannot be used if application is unable to prove some mandatory information. When creating VariableTypes, the subtype has to have the same data type as the supertype. In OPC UA multiple-inheritance is possible but there are no rules for that. Because of the complexity of multiple-inheritance it should be avoided. (Mahnke, Leitner & Damm 2009)

OPC UA has simple and complex ObjectTypes. Simple ObjectTypes only define semantics of the object while complex types define the structure. The ObjectType should be complex if the type is going to have multiple instances that have the

same structure. This is the typical case for a device. Simple types are needed when the instances have different structures, like in an object representing a factory area for example. (Mahnke, Leitner & Damm 2009)

If an object provides only one variable, like a simple sensor for example, using VariableType might be considered. However, this doesn't support extensibility well. If you have also more complex sensors, you need to define them as ObjectTypes. Having both ObjectTypes and VariableTypes representing sensors is harder to handle and bad design. Therefore all of the similar devices should be the same kind of type to enable similar handling. If there are only simple sensors VariableType can be used but if there are more complex sensors, all the sensors should be ObjectType. (Mahnke, Leitner & Damm 2009)

4.3.3 Naming of new types

Common naming practices help programmers to get an idea what a type is doing without reading the whole code. The type libraries created should be named similarly as the predefined OPC UA type libraries. They are called ObjectTypes, VariableTypes etc. They always start with a descriptive name written with uppercase letter and end with Types. Own type libraries have to be named accordingly. Also the types themselves in the OPC UA specification start with a descriptive name written with uppercase first letter and end with the word "Type". Therefore also the own types should start with an uppercase letter and name and end with "Type". An exception is reference type that doesn't have a specified ending.

4.4 Information models in an OPC UA server

Since there are different OPC UA servers there is no generic way of uploading models. The models can be uploaded in different file formats and the nodes of the address space can be defined before uploading the models or dynamically in the server. Also there are different ways to bind the process data to the model. Instead of trying to describe the process of setting up an OPC UA server, this chapter studies the different file formats that can be used to describe OPC UA information models and how the model should be validated before moving it to the server. First the benefits and drawbacks of the file formats are discussed from

a few viewpoints. The viewpoints are capabilities of describing the model, the interchangeability and the possibilities of dynamically changing the information model.

4.4.1 File formats

Information models or the standard nodes of them should be stored in some machine readable file format. This enables populating servers Address Space automatically from the file. However, some constraints always need to be defined in text format. (Mahnke, Leitner & Damm 2009) Having a common format that is both human- and machine-readable allows the model to be processed in future also. It also supports interchangeability of the model. (Postół 2015) OPC Foundation provides the standard nodes and also companion specification nodes in XML-files based on its own UANodeSet XML Schema. (OPC Foundation 2015d) The benefit of using this standardized schema is that it is compatible with some OPC UA tools, like modelers. Since it is defined by OPC Foundation itself, it can be expected to be the format used in future.

The information model needs another file format for the implementation of the Address Space functionality and the connection to the real world. It needs to be capable of instantiating Address Space at runtime. This code is usually generated with some compiler that can be used as an individual solution or embedded to the model designer tool or to a server. (Postół 2015) OPC Foundation provides an open source Model Compiler to generate ANSI C or C# source code from XML files. (OPC Foundation 2016a) The provided source code can be used to instantiate and interconnect nodes at runtime. In addition to the code languages mentioned also other languages like C++ and Java can be used. The selection depends on the server. In this thesis we are interested in developing information models for a .NET server. From now on, the discussion will focus on .NET servers.

The way most of the .NET servers function is the one described above. They load the XML and the code generated. The XML files contain the node set while the code contains the connections to the data. The code handles creation of the instances. However, there are also optional ways that will be discussed next.

Information models can be also specified only with code, like C#. The code can be separately compiled and moved to a .NET server using Dynamic-link library (DLL) -files. As said earlier, the code is needed to provide functionality to the information model. This functionality doesn't only mean methods but also functionality in assigning values, checking ranges etc.

Information models cannot be described only with XML since currently there is no used technique to provide functionality that way. However, in the future new XML scripting techniques might provide functionality to XML-format. For example Stuart (Stuart 2009), has written a patent of an XML based scripting language. Currently there is no de facto XML scripting technique. Of course own methods to do this can be defined but it would require more work and the interchangeability benefits of XML would then be lost. It also cannot be said for sure that there is ever going to be de facto XML scripting standard.

Compared to coding XML has one practical benefit. XML is designed to present structure while code is not. Interpreting structure from code even if it is object-oriented requires more effort. The hierarchies in XML are also more human-readable.

XML and generating executable code from it in server has also another benefit. It allows changing information models at runtime since the XML models can be easily removed from server. However, dynamically changing the information models requires some means of modifying the existing node instances and to maintain their data bindings.

Loading new DLL-files to a server is also possible but a lot more complex than loading XML-files. This is due to the fact that there is no way to unload an existing DLL-file from a .NET assembly. The problem could be solved by loading DLLs to a separate new AppDomain. AppDomain means an isolated layer or environment where applications execute. (Microsoft (c) 2016) When the DLL-files need to be updated, the old AppDomain can be disposed and a new one can be created. (Holstad 2007) The information model is updated by loading the new DLL-files to the new AppDomain.

Of course for the update to be useful also the existing instances of the modified objects have to be changed. One way of doing this could be creating new

instances of every existing instance and creating a method to copy each value of the instance to the new instance. After this the old instances could be removed.

Writing information models with C# and uploading them to server as DLL-files was selected to be used in this thesis. The reason is that functionality still requires some parts of the information models to be presented as code. XML scripting is not developed enough to be used in this yet and it might never be. Dynamically changing the information model database with DLL-files is hard but not impossible. The drawback of the selection is that the readymade tool solutions like modelling tools and servers usually support XML the way described in the beginning of chapter. Also the code doesn't present the structure in an as human-readable format as XML does. Reading and understanding how the XML models are formed and how one should write information model in XML is a lot easier than understanding the code. Programming the models requires much more knowledge of OPC UA, the server functionality and of course programming.

4.4.2 Architecture

The information models logically build upon each other when they are created by programming. A logical architecture is presented in Figure 24. The vendor specific models on the top layer inherit the lower layers and so on. The methods are called directly from the objects. There are a lot of advantages in this kind of architecture. There is a lot of reuse of the lower level components since all the top level components can use the same base. Adding new components is easy and therefore the application is scalable. However, the architecture also has disadvantages. The clearest disadvantage is that changes to the lower levels usually affect the top levels. This kind of architecture is not usually optimal performance vice either because depending on the internal server architecture the communication might need to pass many layers.

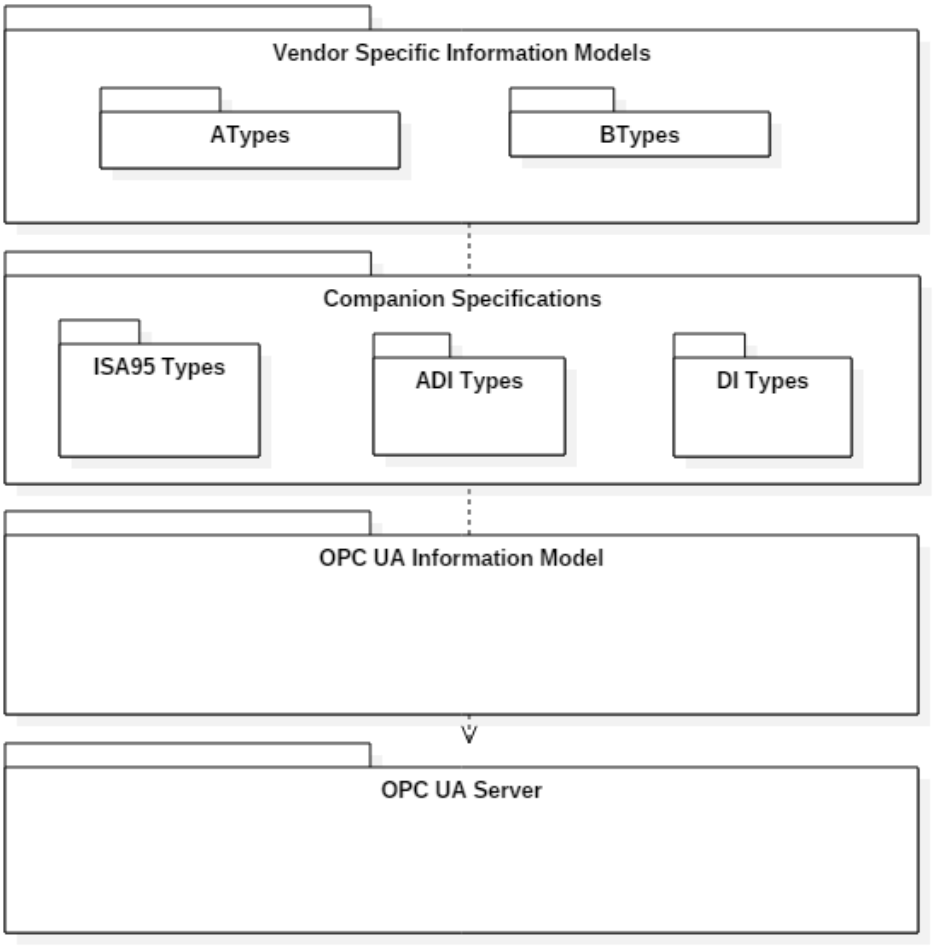

Figure 24. Layered information model architecture.

Another option for the design would be implementing an interface through which all the communication would pass as presented in Figure 25. The idea is to base logic on the interfaces instead of the internal implementation details of the objects. An interface of an object can be thought as a type definition of an object. It defines the methods the object has but the object class itself has to take care of the implementation. Using the interface reduces the dependencies of the code. Because the interface defines the interactions between the object they cannot reference each other directly anymore. Of course having this kind of single package to handle all the communication introduces a single point of failure and a bottleneck. Also the interface package can become really complex if not designed properly.

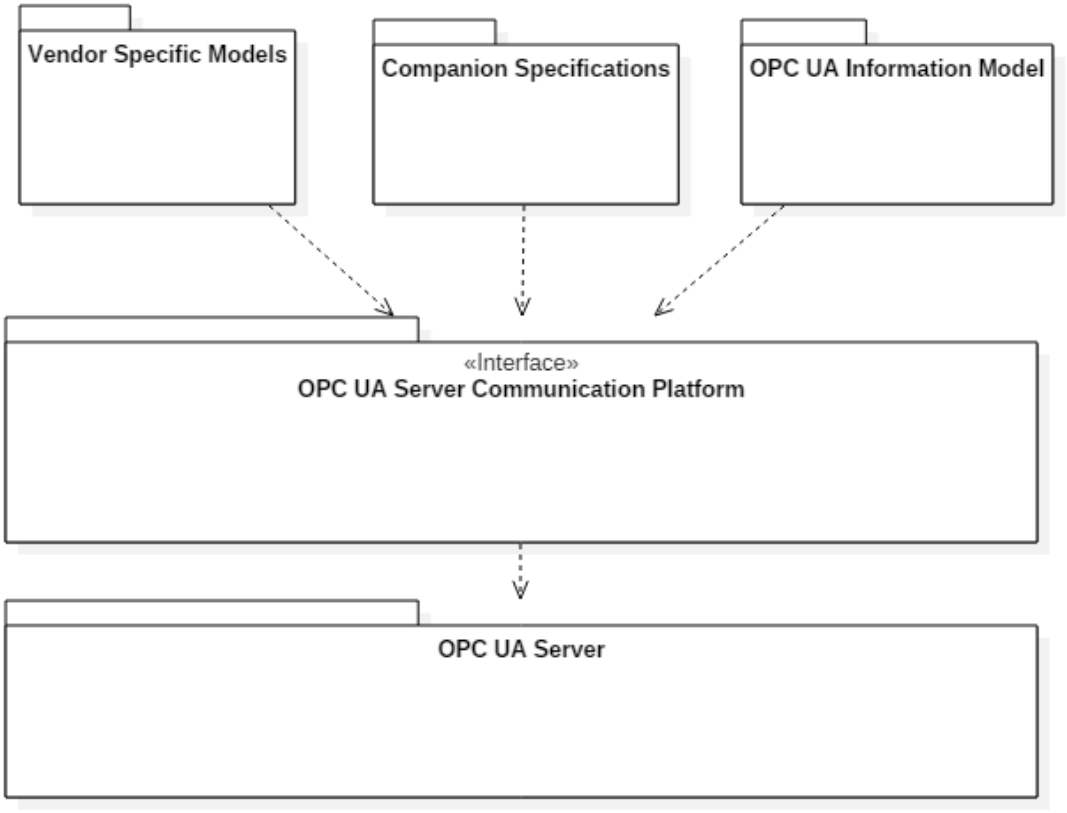

Figure 25. Alternative architecture approach with interface.

The use of interfaces makes sure that there are no references from code instance to another. Because the address space of an OPC UA server builds the references between the nodes using OPC UA reference definitions, referencing objects in the code in unnecessary. If objects reference each other in the code removing the nodes becomes complicated. In addition to removing the node and its OPC UA references from the server, the possible code object references to the object need to be removed. If a code object reference to the object stays, the .NET garbage collector will not remove the actual object. Also recreating the internal code object references when the server is restarted is harder than simply recreating the address space.

The layered approach is definitely more instinctive solution since it is based on the OPC UA Information Model hierarchy. The hierarchy and the inheritances can

be written directly to the code. In the single interface approach, the hierarchy has to be built run-time and not by referencing the objects in the code.

One problem in the layered architecture is that always when an instance of a class is not being used anymore, the node might be removed from memory while there are still some references to it. Because of the objects referencing each other directly there are a lot of references and deleting all of them is hard. With the interface approach this is not a problem.

Another benefit of having an interface for the communications is that the actual information models can be easily modified and new information models can be added. This is because the information models are not aware of each other. Only the communication interface might require modifications when something is altered or added.

In this thesis the architecture the information models are based on is layered architecture. This is because the existing models are using layered architecture. It would require a lot of changes and refactoring to start modelling with different architecture.

4.4.3 Validation

Information models that are not properly done can cause problems in the server. Since programming and modelling tools provide ways to detect syntax errors they are not usually the reason. Logical errors like mistakes in calculations or resource management cannot be detected by programming tools. Of course good practices of coding and clear instructions help avoiding these errors but it should be never trusted that the code is completely error free.

Before a new model is used some validation and checking for the most common logic errors should be done. Having the code or at least the functional parts inspected visually by a colleague could help but isn't enough.

In addition to peer inspection some kind of testing has to be done. There are software applications for continuous integration. This means applications that are capable of building and testing software continuously and even delivering them. The basic idea is that when changes are added to the version control the

software running on the version control server automatically builds the software project. Then it executes the unit tests made. It is necessary to test that the new information model is compatible with also the previous server and information model library versions. If it is not, some feedback is needed to know which versions can be used with the information model. Since unit testing is time consuming and often neglected it is important that it can be done automatically. At least the most critical parts of the system, meaning the ones with possibility to cause the system to fail, should be tested. All kinds of functions or methods with calculations, casting or conversions can be considered critical. Boundaries could be set for example to calculations such as division, square root and logarithms since they cannot be always calculated and cause non-numerical values.

When the information model is delivered to the client's system a sandbox environment is required to allow the developers to safely test the system before launching it. Sandboxing means a technique where the program is isolated from other programs to test it. Often it is a virtual machine that is a copy of the actual server with an identical database and environment. It protects the actual server because it allows only the virtual machine to fail. The reason this is needed is that there is always the possibility that something in the client's system has been changed after the first delivery or during it. Sandboxing takes time especially in the process environment since all the anomalies don't occur in a day. However, it is worth doing at least until a certain predefined level of confidence is reached. How long the sandboxing should be done depends on the nature of the process.

5 INTRODUCTION TO THE EXPERIMENTAL PART

In this chapter the aims of the practical part of the thesis are explained. First the problem with interoperability in a current solution is explained. After that the possible improvement is presented together with steps to achieve that. Also targeted ideal functionality of the solution is given. The second subchapter presents the equipment modelled in the thesis to give a clear image of what is going to be done. It defines specific requirements for the information models to achieve the goals of the practical part.

5.1 Development needs

The aim of the practical part of this thesis is to create information models that enable interoperability between different software. The models are created to Neste Jacobs Oy's NAPCON solution environment. The specific target is to achieve interoperability in NAPCONs distillation column calculation software.

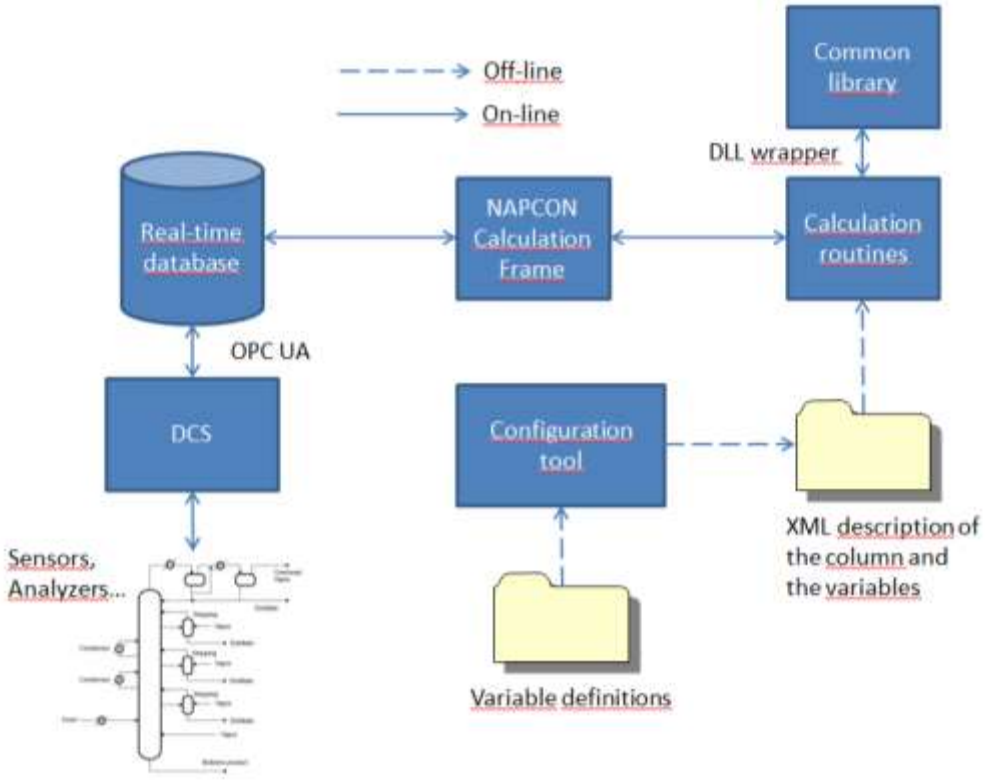

Figure 26. Architecture of the current solution. (Räisänen 2014)

Currently, whenever calculating something with the distillation column calculation tool, the distillation column has to be defined and configured with an offline tool. The tool then creates an XML file. The XML file contains the model of the distillation column and tags of the data needed from the database. When the calculation is started the XML is loaded and de-serialized into instances. The database variables are saved into their own table. The architecture of the current solution is presented in Figure 26.

To improve the interoperability and the flexibility of this solution the information model should exist in the OPC UA Server. The resulting architecture, presented in Figure 27, would be simple and more intuitive than the old one. With information models existing on the server the configuration tool could function online and the calculator could directly use the information models and the data from the database. Also the result could be returned directly. When the information models exist on the server there is no need for external configuration

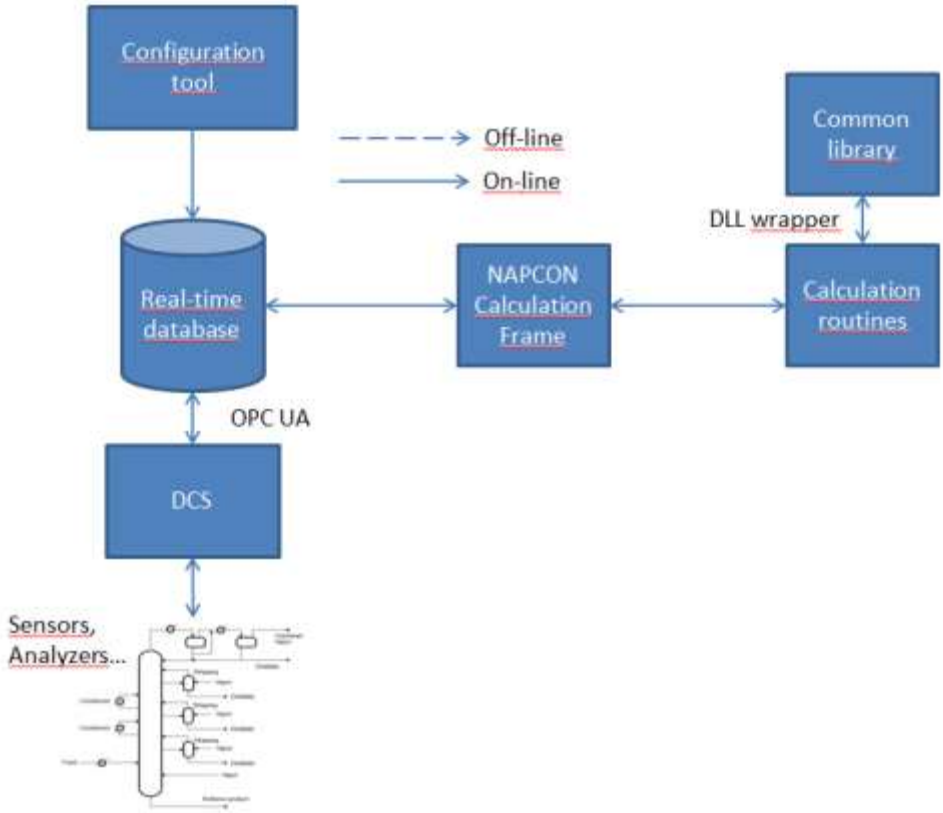

Figure 27. Architecture of the solution with information models in the database

files. The client can see the defined distillation column on the server and the calculations made are therefore more transparent. Also the architecture supports the IoT way of doing things. The models are not only accessible to the calculation tool but also to other clients.

To implement this, first the distillation process is studied as well as the requirements of the calculation. An information model of the distillation column is created using OPC UA information modelling specification. The development started with creating a generic distillation column information model, moving on to the data specific to this application. The model is then moved to the server and its functionality is tested. Some changes are needed for the existing calculation tool to support the model. Also a new or modified configuration tool is needed to

create the instances of distillation columns to the database. However these are not in scope of this thesis. Only the information models are assessed.

5.2 Equipment modelled

The equipment to be modelled in this thesis is a continuous multi-product distillation unit. In addition to all the parts of the column also the feeds and product streams are modelled. Base type definitions are created for all distillation columns to create a good and reusable hierarchy.

In multi-product distillation the feed consists of multiple components. The components are separated in a single column producing multiple products. Multi-product distillation is more complex than a binary distillation column. (Räisänen 2014) Both of these have feed flow coming from the middle of the column. The liquid phase comes out from the bottom of the column. In some cases part of the liquid is vaporized in the reboiler and fed back to the column. The overhead vapor comes out from the top of the column. It is condensed to liquid. Part of the liquid is fed back to the column as a reflux flow and the rest is distillate. The actual distillation columns can be tray columns or packed bed columns.

Multi-product distillation columns have more variety in structure. Mainly the difference is that there are multiple side draws. One possible multi-product distillation column is presented in Figure 28. This column has five different products. The side draws are stripped to remove absorbed light components from the distillate and to feed them back to the column as vapor. Vapor can also be fed from the bottom of the column. This lowers the partial pressure of the stripped light components in the gas phase. It increases the yield of the separation. There are also circulating flows to condensers next to the side draws. The condensers in the mid parts of the column reduce the need of energy in the top of the column. In addition they increase the liquid flow below them and affect the concentration of the product flow. (Räisänen 2014)

Because of the complexity and variety of multi-product distillation columns the information model made from the column has to be abstract and flexible. It has to enable creation of different column structures. The amount of distillate flows out can vary. Columns can have different number of trays. Sometimes there is no

stripping but the vapor flow out from the mid part of the column is condensed directly into liquid and divided into reflux and distillate. In addition to this there can be some kind of thermal integration both between the flows of the column and between multiple different process units. For example in crude oil distillation the feed of the column is heated with the condenser of the top of the column, reflux flows and a separate heater (Räisänen 2014).

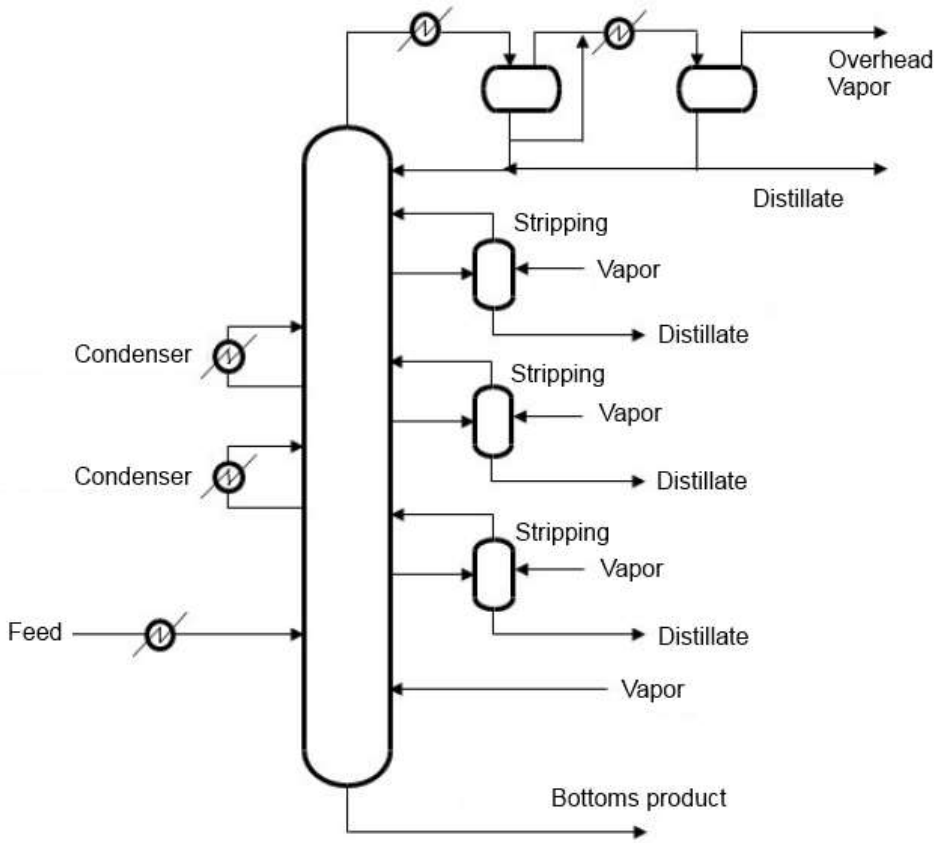

Figure 28. Multi-product distillation column

In this case also knowledge about the calculations and variables needed for them is necessary in addition to the knowledge about the column. In the calculation software the column is divided to unit columns. The unit columns consist of the actual column part, the side draw-off unit with possible stripper and a condenser circulation. It is important to know with which tray the streams are associated.

The bottom and the top unit columns are different from the others since they do not have the same process units related to them.

The column itself has multiple measurement points for temperature, pressure and flow. These are all associated with different trays. Calculating the flows between the trays is important and therefore also the measurements and knowing their location is important. In addition to these there is some additional data about the column, like the pressure loss in trays or the dew point.

The heat exchanger objects like condensers and boilers have information about flows going in and out and also the power used and the maximum power available. These can be used for energy calculations.

Information is required about the flows both inside the column and going from the column to different equipment. Flow properties such as mass flow, pressure and temperature are necessary as well as information about heavy key and vapor fractions and density. These all should be available for the calculations. There are also properties that are calculated like enthalpy, molar flow and molar weight as well as the Watson characterization factor. If available, also the distillation curve temperatures are necessary information.

6 THE MODELLING PROCESS

This chapter explains how the modelling of distillation column was done. First the choice of tools and programming languages are motivated quickly. After that the actual process of modelling is presented. It includes justifying the choices of companion specifications and explaining how they were used.

6.1 Tools used for modelling

The modelling in this thesis was done by programming in C# language by using Visual Studio as the editor. The reason for the choice was that the modelling tools couldn't produce code suitable for the NAPCON OPC UA server. Also programming would have been needed anyway since the modelers are not capable of implementing functionality. The choice to write the models only as code and not using XML makes the model structure a bit harder to interpret for human. Also currently no modelling tools can be used for the produced code but there is a possibility to develop own OPC UA modeler in the future.

The information models created were uploaded to NAPCON UA Server. Browsing the server's address space and creating the node instances was done with NAPCON Information Manager. All the testing and validation was done by examining the models and using the methods of the nodes with Information Manager.

6.2 Development of the base information models

ISA-95 companion specification defines information model that includes for example EquipmentType. Using this ready-made type seemed reasonable because equipment were modelled. Another option would have been creating own type definitions. Using information model from a ready-made companion specification supports interoperability of the model. Also other companion specifications could have been used but ISA-95 seemed the best since it is suitable all kinds of processes. The ISA-95 model for OPC UA has been released already in 2013. (OPC Foundation 2013b) ISA-95 is abstract model for functions

and operations of a plant and therefore suitable for models used in production control systems.

Another companion specification that was used was the Device Integration - specification (DI). The DI model provides a base for modelling devices such as temperature or pressure meters and simplifies the integration of different devices. The DI model has TopologyElement as its base type. It contains parameter and method sets organized by FunctionalGroupType. The DeviceType inherits TopologyElement and is the base type for devices. (OPC Foundation 2013a)

The DI information model had already been programmed to the used type library so there was no need to edit them. The ISA-95 model was not defined so it had to be made. It was programmed following the OPC UA for ISA-95 Common Object model specification. The properties and references of each needed type were read from the specification and transformed into code. The programming was carried out by starting by modelling the EquipmentType and creating all the other types it needed. The new types were then modelled and the associated types created and so on until there were no more associations. As the result whole model was not programmed but only the parts necessary for this work. They were the equipment, physical asset and material information parts. Also ISA-95 Base Information model was required. ISA-95 model can be seen in Figure 29 in OPC UA notation. In the Common Object Model the Role based equipment information, Physical asset information and Material information were the parts modelled.

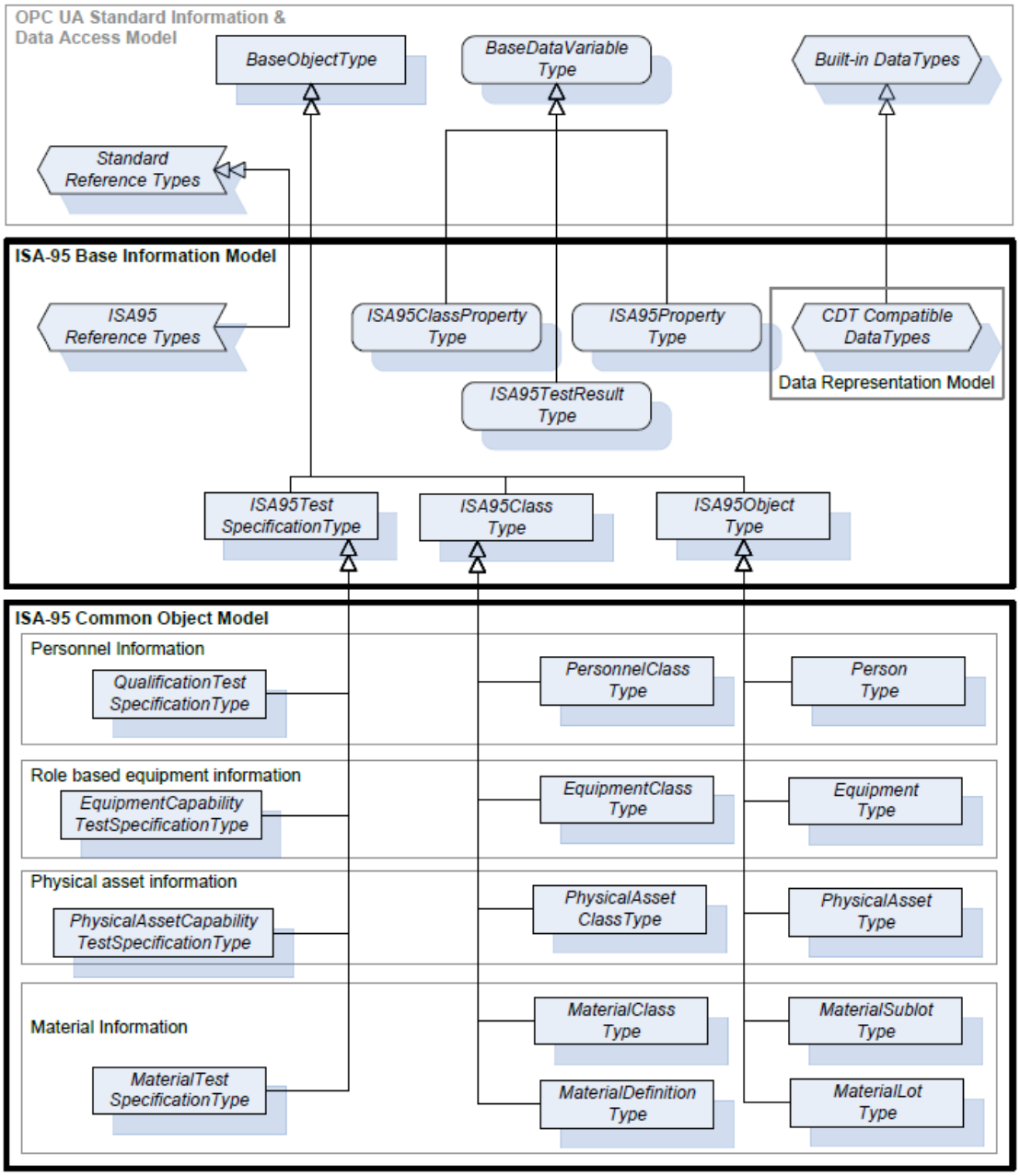

Figure 29. ISA-95 OPC UA Information Model (OPC Foundation 2013b)

Even from the modelled parts some features of the ISA-95 specification were left out on purpose since they were optional and weren't useful for the data of the NAPCON solutions. For example EquipmentCapabilityTestSpecificationType and other similar types were not used because currently there is no need for such capability test data. If the data is needed later the classes can be added. Also Data Representation Model was left out since there was no possibility to create new data types to the server. Instead the most suitable OPC UA data type was selected when there was a need for ISA-95 specific data type.

Before modelling the distillation process some generic process equipment types were created. These were added to their own information model library to be easily subtyped in other information models. ProcessUnitType was created to be a generic base type with common properties for all process units. PipeType was created to be a generic type for all pipes in processes. Both of these types were defined as a subtype of EquipmentType. Also other generic equipment types such as TankType or HeatExchangerType were modelled as a subtype of EquipmentType. To model the flow of fluids from a unit to another FlowsTo reference type was added as a subtype of NonHierarchialReferences. FlowType was created to store the information about the liquid flowing in a pipe. MaterialClassType for ISA-95 specification was used as a supertype of FlowType. To define the relationship of a pipe and a flowing liquid HasMaterialFlow -reference was created. In addition types for equipment, like condenser, stripper and boiler, common for many processes were added.

All in all the modelling process was carried out starting from the highest level of abstraction and proceeding towards the smallest concrete pieces of equipment and devices. First the process unit level was modelled. The distillation process was further on divided into small sub-units, like the column containing trays or a condenser circulation unit containing draw-off and reflux pipes. The equipment were modelled after the sub-unit were done. First the focus was in creating a generic and flexible model of a distillation column and not so much on providing the necessary details for calculations. Because of this approach the model can be used in other applications also and is not limited to the calculation software only.

6.3 Modelling of the distillation column

The modelling process began by studying different multi-product distillation column structures and trying to find the parts common for them. The information was gathered from articles as well as by asking from experts. Also the calculation software related to the distillation columns was studied and the future of it was discussed with experts. A generic model of distillation column was build. The column was divided into small pieces with the idea to keep the model as flexible as possible. Modelling the outline of the process was rather simple but some things like functions for adding trays and decisions about the structure was more involved. It was also time consuming to study how the server and its OPC UA address space works.

After the outline of the distillation process was ready, it was tested and validated with NAPCON Information Manager. The testing was carried out by creating a distillation column with different sub-pieces and trying all the different functions and checking if correct references and nodes exist after that. Also shutting down and restarting the server was tested to see if the nodes and references in the address space are correctly reconstructed after that. The reconstruction was proven out to be somewhat problematic. It had to be taken care that the reference is added actually to the address space not just to the code object so that it would be stored and reconstructed after restart.

The code of the distillation column calculation software and the XML-file used for configuring the software were studied to find out the required data. The measurements in the column and pipes were modelled according to the calculation software's objects. It was found out that also data about the flows, such as distillation curve related data, is required.

Finally to prove the concept of interoperability the information model was connected to data located on another server. Further on this data was used in calculations and results were returned to the database. Also writing data from the local server to another server was tested.

7 IMPLEMENTATION OF THE INFORMATION MODELS

This chapter will first present the generic design of the created information models. First an overview is given about the libraries created and how different types are divided into libraries. Also the structure of the distillation column models is explained together with the idea behind it. After that the thesis contains discussion about the type definitions of the most important equipment and the data the definitions contain.

7.1 Design of the models

The created type definitions were divided into two different libraries. Most of the modelled equipment is only related to distillation processes and therefore located in the DistillationEquipmentTypes -library. However some common equipment was modelled for applicable processes. Type definitions such as PipeType or HeatExchangerType were placed into the ProcessEquipmentTypes -library. The idea is that the ProcessEquipmentTypes library can be used when modelling other process units as well. This promotes code reuse.

Most of the type definitions in ProcessEquipmentTypes-library can be used directly since they are made for common equipment. Notably there is also a ProcessUnitType that should be used as a base for all the process units modelled. The DistillationProcessUnitType is the highest level of abstraction in the distillation process model. It is inherited from ProcessUnitType. ProcessUnitType contains location information and is again inherited from ISA-95 EquipmentType

In general most of the created type definitions are inherited from EquipmentType. A couple of exceptions are type definitions for measurement instruments that are inherited from DI-model's DeviceType. The FlowType defining material flowing in the pipes is inherited from ISA-95 MaterialClassType. In addition to these there are also reference types that were created that inherit the most suitable reference type from the OPC UA -specification.

The created information model of the distillation process is separated to various different components or subunits. The highest level unit,

DistillationProcessUnitType, is an abstract object type having generic components of a distillation process as children. The components are FeedUnitType, BottomUnitType and TopUnitType as well as the DistillationColumnType itself. The UnitTypes present collections of equipment that can be handled as a single component of the bigger process. In the distillation column case they contain at least a pipe connected to the column. There is a possibility to add equipment like a boiler to the bottom unit or a heater to the feed unit.

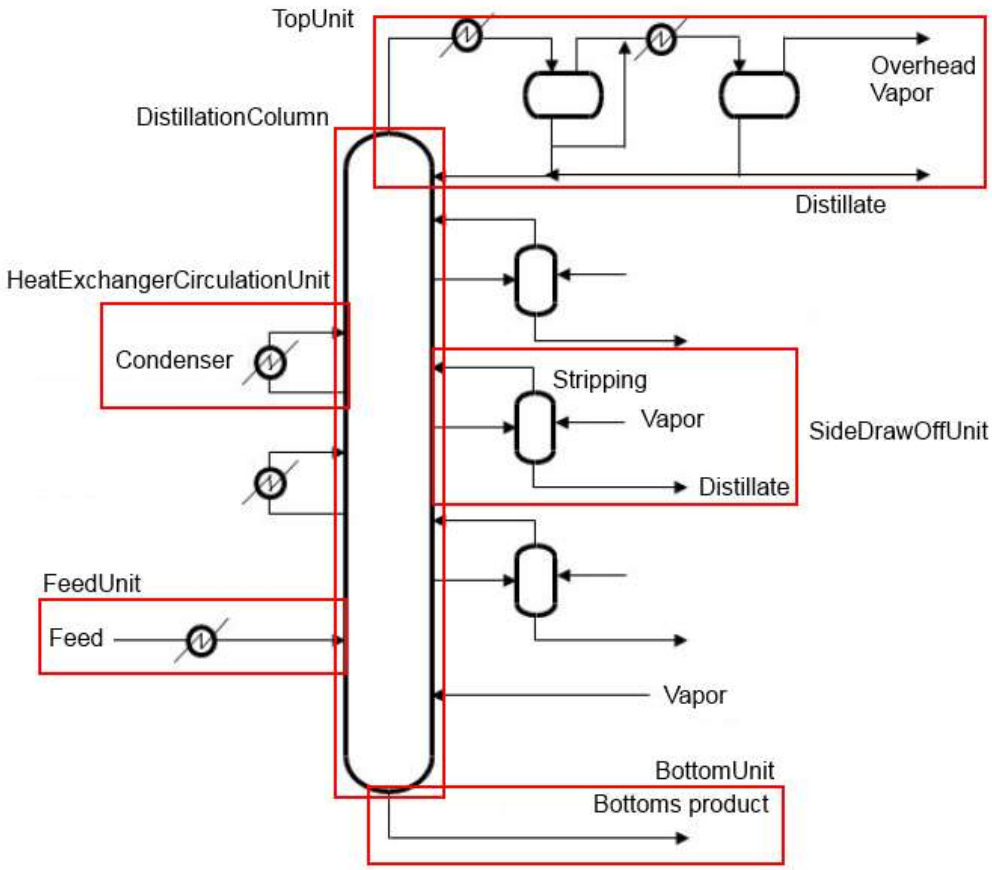

Figure 30. The distillation process divided into components.

The concrete types for distillation processes are MultiProductDistillationProcessUnitType and BinaryDistillationProcessType. Only the first is required for this thesis. In addition to the inherited components the

MultiProductDistillationProcessUnitType has HeatExchangerCirculationUnitType and SideDrawOffUnitType as components. Figure 30 presents how the distillation process is divided to different parts in MultiProductDistillationProcessUnitType. In the figure, the red squares present a sub-component. One column can have multiple condenser circulations and side draw-offs. Figure 31 presents the UML diagram with the main classes of the information model.

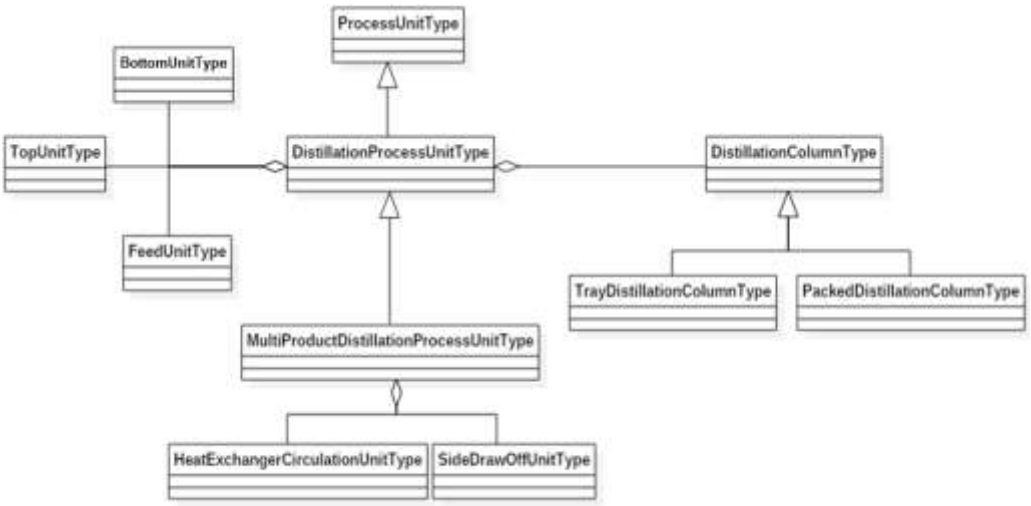

Figure 31. The UML diagram of the main classes of the information model.

7.2 Implementation details

Most important part of the model is the column itself. The DistillationColumnType is an abstract type and cannot be used directly. It has node sets to hold temperature, pressure and flow measurement nodes as children. The concrete subtypes of DistillationColumnType are TrayDistillationColumnType and PackedDistillationColumnType. PackedDistillationColumnType has PackingMaterialType as a child. TrayDistillationColumnType needs to have a set of trays inside and method for adding the trays in order. In addition to these the TrayDistillationColumn has data for the calculations, such as the dew point. There is a TrayType to define one tray. It contains information about the trays above and below. Two special references, VaporFlowTo and LiquidFlowTo, are created to define the relationship between a tray and the upper tray or the tray

below. TrayType also has information about the vapor flow and the liquid flow going through the tray. FlowType is used to define a flow. TrayType has data about its pressure loss to support the calculations. It also has a diameter so that the geometry of the column can be defined. The UML diagram in Figure 32 presents the DistillationColumnType and its related types.

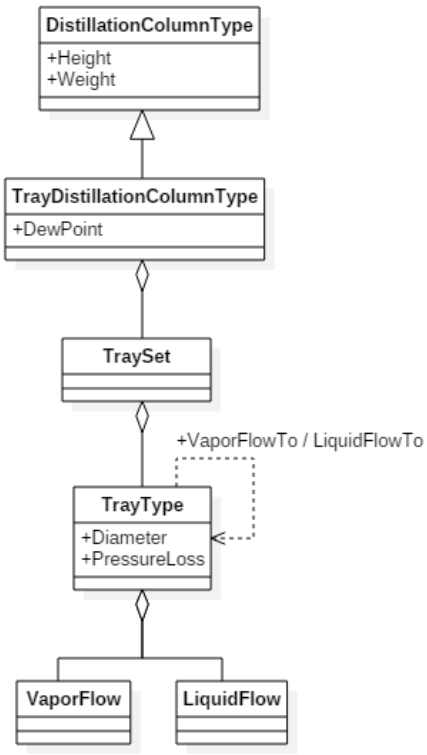

Figure 32. The UML diagram of the DistillationColumnType and related types.

FlowType contains information about the material flowing in a pipe. Things that can be measured or are usually measured from the pipe are children of the PipeType. FlowType contains data that is not measured but known from lab results such as density. FlowType belongs to ProcessEquipmentTypes since it needs to be available for other applications also. However it has a subtype called DistillationFlowType in DistillationEquipmentTypes. The subtype has distillation specific data like the Watson characterization factor or the dataset to store distillation curve temperatures.

PipeType is important since it creates all the connections between other equipment. It uses FlowsTo reference to tell the direction of the flow in the process. It also has information about the material flowing in the pipe and the possible measurements. FlowType is referenced with HasMaterialFlow reference. Figure 33 shows the class diagram of PipeType. Commonly some measurement devices are connected to PipeType but there is no standard measurement set defined.

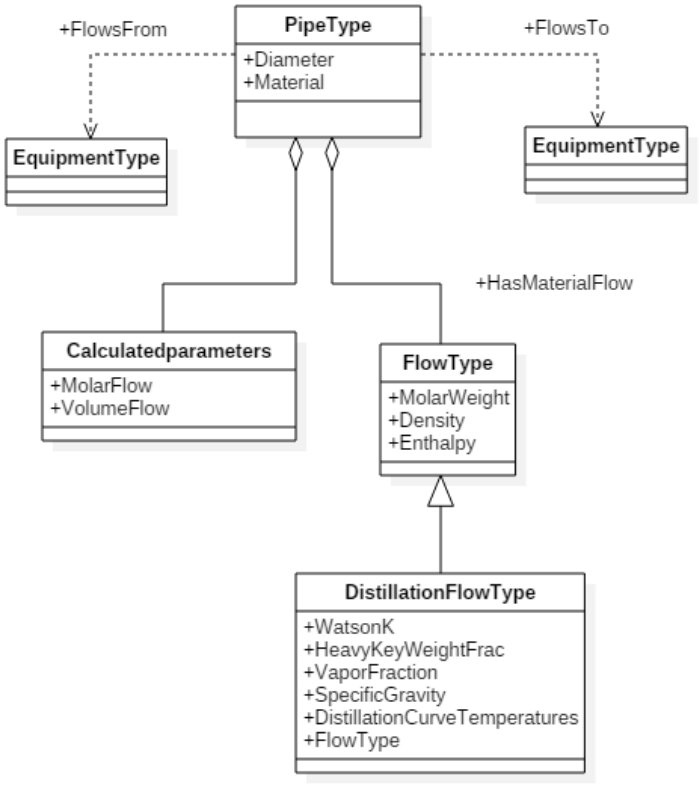

Figure 33. The UML diagram of the PipeType

All the equipment have their own type definitions. For example boilers have BoilerType. The BoilerType is inherited from HeatExchangerType. The type definition for all heat exchangers has data item nodes containing the information how much power it uses and what is the maximum power.

The unit types connected to the distillation column always contain at least the pipe connected to the column. FeedUnitType has optionally also a heat exchanger to warm up the column feed. The TopUnitType has the option to add a condenser and reflux drum and all the pipes connected to them. There can be more than one condenser but most likely these cases can be simplified and modelled with only one condenser. If necessary, there is a possibility for additional condensers. The BottomUnitType allows adding reboiler and pipes related to it. The unit types have methods for creating all the child nodes mentioned here. Of course adding other child nodes and different equipment is possible. The main idea in having these methods is to make sure that the correct set of equipment is added and that the correct references are used between the equipment.

As mentioned earlier the MultiProductDistillationProcessUnit has two additional unit types as children. It also has methods for adding them. HeatExchangerCirculationUnitType always consists of an outflow pipe from the column, a heat exchanger and a reflux pipe. The heat exchanger is HeatExchangerType. Also subtypes like CondenserType can be used in the circulation unit. The SideDrawOffUnitType has always a pipe out from the column. There could be also a stripper, a reflux pipe and a product pipe from the stripper. The SideDrawOffUnit has method for adding a stripper. The PipeType nodes of the condenser circulation and the side draw off are connected to the TrayType nodes in the distillation column. The distillation column can be divided into unit columns for the calculation software with this information. It was decided not to create the unit columns to the information model since the information model is supposed to be generic and applicable for other use cases as well.

Measurement devices have their own type definition called MeasurementDeviceType. The type definition contains an AnalogItem that contains information about the measurement and its value. MeasurementDeviceType has enumeration variable to define if the measurement is pressure, temperature, flow, density or level measurement. The nodes of MeasurementDeviceType can be added to nodes of PipeType or TrayType for example.

A ControllerDeviceType was also created. It has node of MeasurementDeviceType as a child. In addition to that the controller needs to have a set point and an output. These are added as AnalogItems. The type definition of the controller device is really simplified. When creating it the model of ControlEquipment in "OPC Unified Architecture for AutomationML" -specification was studied. However the model was seen as too specific since it represents the controller from the level of the actual device. For this model only presenting the data was important and the implementation of the actual device could be skipped. Therefore it was decided to keep the type definition as simple as possible. (OPC Foundation 2016b) The ControllerDeviceType and the MeasurementDeviceType can be seen in Figure 34.

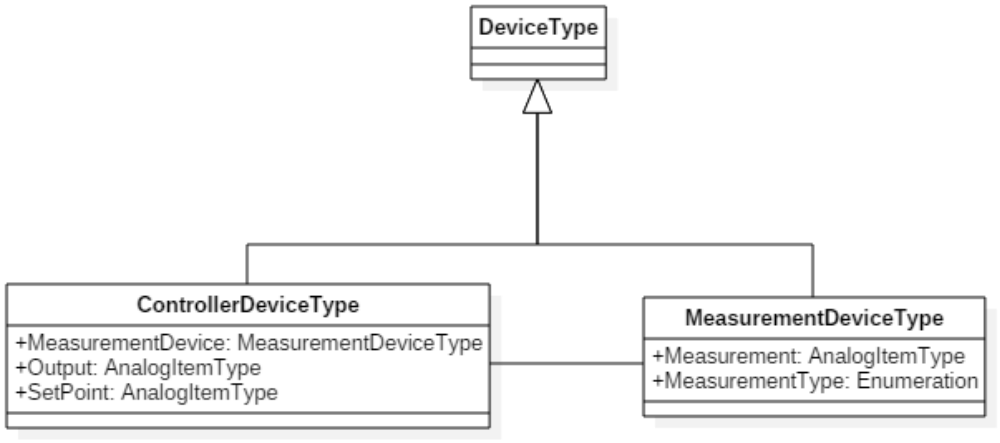

Figure 34. The ControllerDeviceType and the MeasurementDeviceType.

7.3 Connecting to the data

MeasurementDeviceType and ControllerDeviceType type were connected to data existing on another server to prove the interoperability. First the server was connected to the remote database containing the process values. The references could be made once the remote database was connected and accessible.

In the MeasurementDeviceType the measurement was connected to another property using HasInput-reference. The MesurementDeviceType had to be programmed so that adding HasInput-reference triggers an event of writing new

value to measurement and adding value listener to the source node. The value listener was programmed so that every time the value of the source node changes also the measurement changes.

In addition of getting measurement values the ControllerDeviceType had to be capable of writing the output value to the remote database. The same HasInput-reference was used also in this but it was inverted meaning that the output is the input of the remote value. In OPC UA all references have an inverted meaning. Instead of using something called "IsOutput" using the inverted HasInput is OPC UA compliant. Again the type was programmed so that adding the reference triggers function that handles adding value listener to output and writing to the remote property. Therefore every time the output is updated also the remote property will be updated.

The calculation framework can be connected to the information model without any additional modifications to the existing NAPCON Calculation framework. The old distillation calculation tool however cannot be directly used with the information models. It requires some modifications so that it is capable of reading the column structure from the information models. A simple sum calculation was tested with the NAPCON Calculation framework to validate the connection.

With these modifications adding a fully functioning controller would be possible. The set point can be fed to the database by the user or by some program. After this the calculations can compare the measured value obtained from the process to the set point. If they are not the same, the output can be adjusted. The output value is again sent to the process to adjust the actuators.

8 CONCLUSIONS

In this work eight different information modelling standards were studied in order to find the one most suitable for the process industry. Out of these OPC UA specification was selected as the one to be used because according to the comparison made it has the widest modelling capabilities. In addition, OPC UA has many companion specifications and broad industry support. For example Industrie 4.0 is relying on OPC UA. It can be concluded that OPC UA is highly interoperable and compatible with other standards as well. Together with OPC UA also ISA-95 and DI companion specification were used. These were selected because they provided a readymade platform for modelling devices and equipment.

OPC UA modelling tools were evaluated before starting the modelling work. A list of available tools was found from the OPC Foundation's website. Only two of them, OPC UA Address Space Model Designer and UaModeler were available for testing. The UaModeler seemed easier to use and came with all the relevant features. OPC UA Address Space Model Designer provides more features such as binding process data to the model already at the modelling phase.

Despite testing the different tools the actual models were created using C# programming language. This was because the modelling tools export the developed models as XML. The OPC UA server used for the information models had the earlier type libraries in C# so it was feasible to use the same method. Also the XML-files would still require code with them to provide functionality.

In the experimental part an information model of a distillation column was developed. The aim was to formulate good practices of information modelling and figure out how the models should be created. Some rules were discovered and generic instructions were given in Chapter 5.3.

When studying possible architectures for information models it was noted that instead of having the models interact with each other hierarchically in a layered form it might be beneficial to handle the communications through interfaces. This kind of architecture would provide flexibility and modifiability to the models.

During the development of information models it was found out that using objects and referencing them from other objects in code is problematic. When the server is restarted and the address space reconstructed the object references made in code won't be correct. Also when removing nodes the object references would still stay there and be wrong after the deletion. The code should be made using interfaces rather than classes. When using interfaces the object references won't be made and only way to use other objects is through OPC UA references. With interfaces the code has less coupling.

It was found out that coding the information models is time consuming and requires some expertise. First the developer has to get familiar with the existing code and understand how the models work. Studying the existing code is required always when a new programmer has to develop information models. Of course the instructions help but some studying always needs to be done to understand the context of the instructions. Time is consumed also when doing the actual coding since the code also has a lot of unavoidable repetition. Creating a modelling tool could be a solution to speed up the production.

The actual model created was evaluated by creating and studying the possible address space with NAPCON Information Manager and by speaking to experts. The structure of the model was proven to be flexible enough to be able to present different kinds of distillation columns. The model was considered suitable for the needs of distillation column calculation software. The model was seen as a good basis for further development. The generic types created and the ISA-95 types can be used as a starting point for creating other information models.

Connecting parts of the model to data and calculations proved that the model can really provide interoperability between different process parts. Creating more information models for different parts of a plant and further on connecting these models to calculation and control software creates new opportunities to enhance the production processes.

8.1 Future research and development

It is recommendable to develop a graphical OPC UA modelling tool to simplify information modelling in the future. This tool would need to provide code directly

suitable for the NAPCON UA Server. Even though the development work requires time, more time is lost in studying the modelling process from the existing models and code. The benefits are that anybody could design and create the information models without much knowledge about how the server works. Programming skills would still be needed to create functionality but not for all the models. The generated code would also be more robust since it wouldn't have any programming mistakes. Of course logical errors would still be possible.

Because of the logical errors a test environment for the models is necessary. First the tests are run in the development environment. It should be studied what are the most common and most critical errors and how they could be tested. Another test related development target is to create a sandbox testing practices. This enables testing the product in the actual runtime environment without the risk of disturbing the actual automation system.

One important aspect of the thesis was if the models can be loaded dynamically or not. Loading type definitions dynamically is possible when using C# and multiple .NET AppDomains. Achieving this requires major modifications to the current server.

The type libraries should be modified so that they use safe server interfaces to define the types rather than interacting directly with other objects inside the server address space. Classes interacting with each other creates rather high coupling to the code, while the interfaces are more decoupled. References in information models should be made using OPC UA address space references only. The classes referencing each other cause problems since the object references stay even if the node related to the object is deleted.

A graphical tool is required to simplify building the address space of the distillation column and to connect it to data. The connections of nodes and the structure of the model are hard to understand with tree based solutions. The tool should be able to create a graphical presentation of the address space and should allow creation of type instances and references to the address space.

REFERENCES

Adolphs, P., Bedenbender, H., Dirzus, D., Ehlich, M., Epple, U., Hankel, M., Heidel, R., Hoffmeister, M., Huhle, H., Kärcher, B., Koziolek, H., Pichler, R., Pollmeier, S., Schewe, F., Walter, A., Waser, B. & Wollschlaeger, M., 2015, *Status Report: Reference Architecture Model Industrie 4.0 (RAMI4.0)*, ZVEI, Düsseldorf, Germany.

Arrowhead, 2016a, *Arrowhead - General Overview* [Online]. Available: http://www.arrowhead.eu/about/general-overview/ [2016, 04/25].

Arrowhead, 2016b, *Arrowhead framework* [Online]. Available: http://www.arrowhead.eu/about/arrowhead-common-technology/arrowhead-framework/ [2016, 04/25].

Atzori, L., Iera, A. & Morabito, G., 2010, "The Internet of Things: A survey", *Computer Networks,* vol. 54, no. 15, pp. 2787-2805, http://dx.doi.org.libproxy.aalto.fi/10.1016/j.comnet.2010.05.010.

AutomationML consortium, 2010, *AutomationML Part 1 - Architecture and general requirements (Whitepaper)*, AutomationML consortium, Magdeburg, Germany.

Bandyopadhyay, D. & Sen, J., 2011, "Internet of things: Applications and challenges in technology and standardization", *Wireless Personal Communications,* vol. 58, no. 1, pp. 49-69, 10.1007/s11277-011-0288-5.

Bauer, M., Boussard, M., Bui, N., Carrez, F., Jardak, C., De Loof, J., Magerkurth, C., Meissner, S., Nettsträter, A., Olivereau, A., Thoma, M., Walewski, J.W., Stefa, J. & Salinas, A., 2013a, *Deliverable D1.5 – Final architectural reference model for the IoT v3.0*, Internet of Things - Architecture.

Bauer, M., Bui, N., Carrez, F., Giacornin, P., Halller, S., Ho, E., Jardak, C., De Loof, J., Magerkurth, C., Nettsträter, A., Serbanati, A., Thoma, M., Walewski, J.W. & Meissner, S., 2013b, *Introduction to the Architectural Reference Model for the Internet of Things (Whitepaper)*, Internet of Things - Architecture.

Breivold, H.P. & Sandström, K., 2015, "Internet of Things for Industrial Automation -- Challenges and Technical Solutions", *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pp. 532-539, 10.1109/DSDIS.2015.11.

Brunner, C., 2008, "IEC 61850 for power system communication", *Transmission and Distribution Conference and Exposition, 2008. D. IEEE/PES*IEEE, pp. 1-6, 10.1109/TDC.2008.4517287.

Burke, T.J., 2013, "OPC Foundation open standards - Automation and business interoperability improves efficiency", [Online]. Available: https://www.isa.org/standards-publications/isa-publications/intech-magazine/2013/december/opc-foundation-open-standards/.

CAS, 2016, *OPC UA Address Space Model Designer*, 3.20th edn, CAS, Poland.

CAS, 2011a, *OPC UA ADDRESS SPACE MODEL DESIGNER -- STANDARD & PROFESSIONAL EDITIONS COMPARISON*, CAS.

CAS, 2011b, *OPC UA Address Space Model Designer Professional (Datasheet)*, CAS, Poland.

Cotton, D., Grissom, M., Spalding, D. & Want, R., 2012, *Standardization Barriers in the Petroleum Industry*, University of Colorado, Boulder, USA.

Drath, R., Luder, A., Peschke, J. & Hundt, L., 2008, "AutomationML - the glue for seamless automation engineering", *2008 IEEE International Conference on Emerging Technologies and Factory Automation,* IEEE, Hamburg, Germany, pp. 616-623, 10.1109/ETFA.2008.4638461.

Federal Ministry for Economic Affairs and Energy, 2016a, *Cooperation between Plattform Industrie 4.0 and Industrial Internet Consortium* [Homepage of Federal Ministry for Economic Affairs and Energy], [Online]. Available: https://www.plattform-i40.de/I40/Redaktion/EN/PressReleases/2016/2016-03-02-blog-iic.html [2016, 09/13].

Federal Ministry for Economic Affairs and Energy, 2016b, *Plattform Industrie 4.0 and Industrial Internet Consortium agree on cooperation* [Homepage of Federal Ministry for Economic Affairs and Energy], [Online]. Available: http://www.plattform-i40.de/I40/Redaktion/EN/PressReleases/2016/2016-03-02-kooperation-iic.html [2016, 04/17].

Fraunhofer IOSB, 2016, *OPC-UA-Modeler* [Online]. Available: http://www.iosb.fraunhofer.de/servlet/is/35891/ [2016, 05/26].

Harju, J.H., 2015, *Plant information models for OPC UA: case copper refinery*, Tampere University of Technology, Tampere, Finland, 62 p.

Holm, T., Christiansen, L., Göring, M., Jäger, T. & Fay, A., 2012, "ISO 15926 vs. IEC 62424 — Comparison of plant structure modeling concepts", *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, pp. 1-8, 10.1109/ETFA.2012.6489662.

Holstad, S., 2007, *Using AppDomain to Load and Unload Dynamic Assemblies* [Homepage of Clarity Consulting], [Online]. Available: https://blogs.claritycon.com/blog/2007/06/using-appdomain-to-load-and-unload-dynamic-assemblies/ [2016, 06/08].

IEEE, 2015, *Define IoT* [Homepage of IEEE], [Online]. Available: http://iot.ieee.org/definition.html [2016, 04/12].

Industrial Internet Consortium, 2016*, Industrial Internet Consotrium - Working Groups* [Online]. Available: http://www.iiconsortium.org/working-committees.htm [2016, 05/25].

Industrial Internet Consortium, 2015, *Industrial Internet Reference Architecture (Whitepaper)*, Industrial Internet Consortium.

ISA-95, 2015, *ISA-95 Enterprise Control Systems - General Information* [Online]. Available: https://isa-95.com/isa-95-enterprise-control-systems/ [2016, 04/26].

ISO 15926-1, 2004, *Industrial automation systems and integration -- Integration of life-cycle data for process plants including oil and gas production facilities -- Part 1: Overview and fundamental principles*, 1st edn, International Organization for Standardization, Geneva, Switzerland.

ISO/IEC Guide 2:2004, 2004, *Standardization and related activities — General vocabulary*, 8th edn, International Organization for Standardization, Geneva, Switzerland.

ISO/TS 15926-11, 2015, *Industrial automation systems and integration -- Integration of life-cycle data for process plants including oil and gas production facilities -- Part 11: Methodology for simplified industrial usage of reference data*, International Organization for Standardization, Geneva, Switzerland.

Johnston, A., 2013, *Oil and Gas Interoperability Pilot & ISO TC 184 OGI Technical Specification -presentation*, MIMOSA, MIMOSA/PCA/FIATECH Conference, San Antonio, USA.

Johnston, A., Cormac, R., Klein, J., Grossmann, G., Bever, K. & Hyre, B., 2012, *Recording of Live Oil & Gas Interoperability (OGI) Phase 1 Pilot Demonstration*, MIMOSA, YouTube (https://www.youtube.com/watch?v=ShvX4_C7QJg).

Johnston, A., Hoppe, S. & Sandmark, N., 2015, *The Open Industrial Interoperability Ecosystem and the Oil and Gas Interoperability Pilot - presentation*, Standards Leadership Council, Houston, USA.

Kagermann, H., Wahlster, W. & Helbig, J., 2013, *Recommendations for implementing the strategic initiative INDUSTRIE 4.0*, Platform Industrie 4.0, Frankfurt/Main, Germany.

Kostic, T., Preiss, O. & Frei, C., 2003, "Towards the formal integration of two upcoming standards: IEC 61970 and IEC 61850", *Large Engineering Systems Conference on Power Engineering, 2003,* IEEE, pp. 24-29, 10.1109/LESCPE.2003.1204674.

Kumar, R. & Bose, A.K., 2015, "Internet of Things and OPC UA", IARIA, Rome, Italy, pp. 38-43.

Lu, Y., Morris, K. & Frechette, S., 2016, *Current Standards Landscape for Smart Manufacturing Systems*, NIST, 10.6028/NIST.IR.8107.

Mahnke, W., Gössling, A., Graube, M. & Urbas, L., 2011, "Information modeling for middleware in automation", *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pp. 1-7, 10.1109/ETFA.2011.6059111.

Mahnke, W., Leitner, S. & Damm, M., 2009, *OPC Unified Architecture,* Springer, Berlin.

McMorran, A.W., 2007, *A Introduction to IEC 61970-301 & 61968-11: The Common Information Model*, University of Strathclyde, Glasgow, UK.

Melik-Merkumians, M., Baier, T., Steinegger, M., Lepuschitz, W., Hegny, I. & Zoitl, A., 2012, "Towards OPC UA as portable SOA middleware between control software and external added value applications", *IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA), 2012,* IEEE, Krakow, Poland, pp. 1-8, 10.1109/ETFA.2012.6489640.

Microsoft, 2016, *AppDomain Class* [Online]. Available: https://msdn.microsoft.com/en-us/library/system.appdomain(v=vs.110).aspx [2016, 06/08].

MIMOSA, 2016, *Open Industrial Interoperability Ecosystem (OIIE)* [Homepage of MIMOSA], [Online]. Available: http://www.mimosa.org/open-industrial-interoperability-ecosystem-oiie [2016, 04/18].

Mitchell, G., 2016, *Standards Interoperability Breaks Silos In Operating Facilities* [Homepage of The Manufacturing Connection], [Online]. Available: http://reliabilityweb.com/articles/entry/standards-interoperability-breaks-silos-in-operating-facilities [2016, 04/08].

Moriarty, R., O'Connell, K., Smit, N., Noronha, A. & Barbier, J., 2015, *A New Reality for Oil & Gas -- Complex Market Dynamics Create Urgent Need for Digital Transformation (Whitepaper)*, Cisco.

Object Management Group, 2016, *Object Management Group and OPC Foundation Announce Collaborative Strategy for the DDS and OPC UA Connectivity Standards* [Online]. Available: http://www.omg.org/news/releases/pr2016/04-13-16.htm [2016, 05/04].

OPC Foundation, 2016a, *GitHub: OPCFoundation/UA-ModelCompiler* [Online]. Available: https://github.com/OPCFoundation/UA-ModelCompiler [2016, 06/07].

OPC Foundation, 2016b, *OPC UA - AML Libraries NodeSet* [Homepage of AutomationML], [Online]. Available: https://opcfoundation.org/UA/AML/AMLLibs/Opc.Ua.AMLLibraries.NodeSet2.xml [2016, 08/18].

OPC Foundation, 2016c, *Unified Architecture* [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/ [2016, 04/26].

OPC Foundation, 2016d, *Unified Architecture - OPC Unified Architecture Specification* [Online]. Available: https://opcfoundation.org/developer-tools/specifications-unified-architecture [2016, 04/26].

OPC Foundation, 2015a, *OPC UA in the Reference Architecture Model RAMI 4.0* [Online]. Available: https://opcfoundation.org/opc-connect/2015/06/opc-ua-in-the-reference-architecture-model-rami-4-0/ [2016, 05/12].

OPC Foundation, 2015b, *OPC Unified Architecture, Part 5 Specification - Part 5: Information Model*, 1.03rd edn, OPC Foundation.

OPC Foundation, 2015c, *UA Overview* [Online]. Available: https://opcfoundation.org/wiki/index.php/UA_Overview [2016, 04/26].

OPC Foundation, 2015d, 08/18/2015-last update, *UA/Schemas* [Online]. Available: https://opcfoundation.org/UA/schemas/ [2016, 06/07].

OPC Foundation, 2013a, *OPC Unified Architecture for Devices - Companion Specification*, 1.01st edn, OPC Foundation.

OPC Foundation, 2013b, *OPC Unified Architecture for ISA-95 Common Object Model Companion Specification*, 1.0th edn, OPC Foundation.

Plattform Industrie 4.0, 2016, *What is Industrie 4.0?* [Homepage of German Federal Ministry for Economic Affairs and Energy], [Online]. Available: http://www.plattform-i40.de/I40/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html [2016, 04/15].

Postół, M., 2015, *OPC UA INFORMATION MODEL DEPLOYMENT (Whitepaper)*, CAS.

Prismtech, 2016, *Industrial Internet Reference Architecture* [Online]. Available: http://www.prismtech.com/products/vortex/technologies/industrial-internet-reference-architecture [2016, 05/04].

Räisänen, A., 2014, *Reaaliaikaisen laskentaohjelmiston arkkitehtuurikehitys ja yleistys monituotetislauskolonneille*, Aalto University, Espoo, Finland, 100 p.

Rohjans, S., Uslar, M. & Appelrath, H., 2010, "OPC UA and CIM: Semantics for the smart grid", *IEEE PES Transmission and Distribution Conference and Exposition, 2010,* IEEE, New Orleans, USA, pp. 1-8, 10.1109/TDC.2010.5484299.

Rouse, M., 2014, *Service-oriented architecture (SOA)* [Homepage of TechTarget], [Online]. Available: http://searchsoa.techtarget.com/definition/service-oriented-architecture [2016, 05/03].

RTI News, 2016, *Object Management Group and OPC Foundation Announce Collaborative Strategy for the DDS and OPC UA Connectivity Standards* [Homepage of RTI], [Online]. Available: http://www.rti.com/company/news/omg-opcf-collaborative-strategy.html [2016, 04/17].

Schleipen, M., 2010, *Automated Production Monitoring and Control System Engineering by Combining a Standardized Data Format (CAEX) with Standardized Communication (OPC UA).*, INTECH Open Access Publisher, 10.5772/9503.

Schmidt, N. & Lüder, A., 2015, *AutomationML in a Nutshell (Whitepaper)*, AutomationML consortium, Magdenburg, Germany.

Schneider, S., 2015, *Data Connectivity in the Industrial Internet Reference Architecture* [Homepage of Real-Time Innovations], [Online]. Available: https://blogs.rti.com/tag/iira/ [2016, 04/25].

Schuller, A. & Epple, U., 2012, "PandIX—Exchanging P&I diagram model data", *IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA), 2012*IEEE, Krakow, Poland, pp. 1-8, 10.1109/ETFA.2012.6489537.

Selway, M., Stumptner, M., Mayer, W., Jordan, A., Grossmann, G. & Schrefl, M., 2015, "A conceptual framework for large-scale ecosystem interoperability", *Conceptual Modeling,* Springer, pp. 287-301.

Slaughter, A., Bean, G. & Mittal, A., 2015, *Connected barrels: Transforming oil and gas strategies with the Internet of Things*, Deloitte University Press.

Standards Leadership Council, 2016, *SLC Webpage* [Online]. Available: http://www.oilandgasstandards.org/ [2016, 04/15].

Stuart, A.F., 2009, *XML based scripting language*, Google Patents.

Unified Automation, 2016a, *UaModeler*, 1.4.3 edn, Unified Automation, Poland.

Unified Automation, 2016b, *UaModeler "Turns Design into Code"* [Online]. Available: https://www.unified-automation.com/products/development-tools/uamodeler.html [2016, 05/26].

van der Linden, D., Granzer, W. & Kastner, W., 2011, "OPC Unified Architecture (OPC UA) new opportunities of system integration and information modelling in automation systems", *9th IEEE International Conference on Industrial Informatics (INDIN), 2011,* IEEE, Lisbon, Portugal, pp. 1-169, 10.1109/INDIN.2011.6035016.

Weyrich, M. & Ebert, C., 2016, "Reference Architectures for the Internet of Things", *Software, IEEE,* vol. 33, no. 1, pp. 112-116.

Whitmore, A., Agarwal, A. & Xu, L., 2014, "The Internet of Things---A survey of topics and trends", *Information Systems Frontiers,* vol. 17, no. 2, pp. 261-274, 10.1007/s10796-014-9489-2.